Universidad
Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería de Sistemas Audiovisuales

TRABAJO FIN DE GRADO

A Multimodal Approach for the Automatic Assessment of Viewer Subjective Perception of Youtube Videos

Fernando Madrid Recio

Tutor: Fernando Fernández Martínez

22th June 2015

Index

List of Figures

List of Tables

Chapter 1

Introduction

Information overload is a common problem for our perception sensors. They generate signals continuously and it would be really difficult and costly to process all the information we get from them. Due this reason, it is important for our nervous system to make decisions of which part of any information has to be prioritized, making a sequential list of the different parts of a visual scene. This eye-tracking process is named as *selective attention.*

Carrying this idea to telecommunications field, makes this matter more relevant and it could be an important aspect concerning with visual data as images or videos. For instance, it might be used to identify which parts of the scene would be prioritized in the visualization of the content, taking this into account when dealing with treatment of data.

The *saliency* concept was introduced in order to express visually the behavior of this process. In order to do so, the image in question is segmented in different *saliency areas* depending on how salient they are. In other words, the image is visually segmented trying to emulate the *selective attention* process.

Many researches have worked with the *saliency* concept, trying to emulate this eye-tracking behavior, implementing different algorithms and improving the results achieved previously. If the selective attention defines which parts of the scene have to be prioritized by the human eye, it would be interesting to study if these parts are important for the subjective perception of the viewer.

The purpose of this research is directly related with this matter. We will demonstrate that *saliency* has an important impact in the subjective perception of a viewer of a scene. It is important to point that saliency is not influenced by any affective response, working this way with a measurable and objective concept which may be used as an approach of viewer subjective perception. To deal with this goal, low-level saliency features will be extracted

1.1 Adopted Domain

This research project is focused on the understanding of how saliency could influence the impression perceived by a viewer of a video. However, this perception cannot be perfectly assessed since some inherent bias is inevitable. This problem is an existing error that should be understood and taken into account in any statistical analysis or experiments. For instance, and focused on this research, the perception of a viewer can be affected by tendencies, inclinations or feelings of the individual (Attentional bias [1]). In order to lessen the impact of this issue it is important to define a simplified research field, limiting tendencies or feelings of the viewers. Due this, car commercials were chosen as the video domain for the study. These and other following choices were previously stablished by *Alejandro Hernández García* in his project called *Aesthetics Assessment of Videos through Visual Descriptors and Automatic Polarity Annotation* [2] which will be named several times in this research.

This car commercial database was extracted from YouTube and was simplified obtaining finally a collection of 138 car commercial videos. This collection will be used for this project too and will be the domain for the proposed computational model.

This way, we will demonstrate the impact of the saliency in subjective perception, focusing this assumption on car commercial videos extracted from YouTube. As consequence of this assumption, saliency also would modify the video ratings in this platform

1.2. Objectives

With the previous saliency knowledge, we attempt to study this concept and its correlation with the perception of the viewer. In order to do so, a computational model will be developed defining car commercial videos as the domain to work with. Thus, the development of this model provides the following study research:

- **Study of the correlation between saliency features and the impression perceived by the viewer:**
  First of all, in [2] some metadata from YouTube was compiled (e.g. views, likes, dislikes) and the videos were labeled into two different classes. In this way, each video belongs to only one class corresponding to either *good* or *bad* videos.
  To study this correlation, several *features* related to saliency will be extracted from the frames of the different videos located in the database. After this, we will stablish for each video a comparison between its saliency features collection and its class in

order to know if this kind of features have an influential impact on the perception of a video.

In that way, a classification model based on saliency will be developed. The development of this classification model will follow these steps:

- First of all, a training procedure will take place. For this step and working with a training collection of videos, our classification model will learn which features and values normally describe a class of video (i.e., for this case, *good* videos and *bad* videos)
- After this, our classification model will be tested, predicting which class a new unlabeled video belongs to. In order to make this prediction possible, the classification model will be based only on the study of the features commented before. The results obtained will be studied and compared using different data sets, as well as different classification algorithms in order to find the best solution. Despite this best solution will not provide the correct classification at 100%, it will improve the results retrieved by a benchmark algorithm, thus demonstrating that saliency has an important statistical influence in this video classification and as consequence, in the perception of the videos.

In brief, the main goal of this project is to purpose a new computational model based on saliency related features for the automatic assessment of viewer subjective perception of a video.

Chapter 2

State of the Art

The main goal of this chapter is to name and explain all the related work previously studied that inspires the basis of this research. These previous researches are focused on the saliency concept, giving an idea of the current state of this matter as well as the algorithms previously implemented. After this brief related with the study field, some applications that require saliency maps, as well as the motivation to do this project will be exposed. The project will be explained as a new proposal which will help to the saliency study.

2.1 Related Work

We are introducing the main contributions to the state of the art concerning with saliency. At this point, it is important to differentiate between static and dynamic saliency approaches, since the techniques and proposals exposed in these researches may differ depending on this matter.

2.1.1 Static saliency approaches

In the introduction of this paper the concept of selective attention was explained. However, what determines which part of the visual scene has to be processed and which ones will be discarded? *Bottom-up* and *top-down* factors contribute to this question. We could stablish that top-down factors are those one which take into account the subjective internal state of the subject that perceives the stimuli and, on the other hand, bottom-up factors do not take into account any subjective internal state or any previous assumption of the content. Subjects selectively direct the attention when visualizing a scene depending on both, top-down and bottom-up factors, an idea that has been studied from long time ago since William James [3] stablished it.

Bottom-up provides a generic approximation of attention and deals with aspects that are independent of any internal state of the subject, being easier to understand and to measure them. This is the reason why the most part of the current researches rely on this kind of model when working with still images, existing several attempts to understand bottom-up attention [4][5][6][7].

Possibly the most influential one was made by *Christof Koch* and *Shimon Ullman (Koch and Ullman,1985)* [8]. They proposed a model in which all the bottom-up visual features that contribute to the *selective attention* process (such as color, orientation, movement) were included in a single global map called *saliency map.* In this way, saliency at a given location is determined of how different this location is from its surround, attending to features like color or orientation.

This study was just conceptual and the actual first implementation of a saliency map was made by *Niebur* and *Koch* (1996) [9] giving a more precise definition: "The purpose of the saliency map is to represent the saliency at every point in the visual field by a scalar quantity studying its spatial distribution". From this new definition and working with the Niebur and Koch framework, several algorithms to calculate the saliency have been implemented, trying to develop an enough accurate saliency map. In this map each pixel is assessed by a single scalar quantity indicating its saliency value.

The saliency map model they proposed was made by color, intensity, orientation and motion measures. Furthermore, they affirmed that a sequential scanning made by our perception sensors was directly influenced by the salience. As explained before, salience map was originally formed by visual features based on bottom-up attention discarding top-down attention influence. Despite its complexity, this issue plays an important role in attentional selection and, as a consequence, it should be taken into account changing the original concept. For instance, previous studies based on attention experiments [10] demonstrated that some objects or elements, such as text or human faces, are naturally salient for humans, being totally independent of the way they are shown in the scene.

For this reason, some top-down influence was introduced in this model. This kind of factors were implemented as additional inputs to the corresponding parts of the saliency map. The authors provide some examples as the task developed by *Posner* [11], who explored a simple model in which some subjects were instructed to attend selectively to one part of the visual space.

This first implementation [9] was improved by a new model described by Itti *et al.* (1998) [12]. This model studies the differences of colors, saturation and orientations on the image in question, segmenting it into fragments depending on these features values. The saliency values assigned to the pixels are dependent of discontinuities of these features. This way, the algorithm studies the relative positions of the fragments, and then attaches a higher saliency value for those that share similar feature values but are isolated. On the other hand, a lower value is assigned to fragments that are close to each other. In other words, the algorithm takes into account dissimilarities in pixels neighborhood, identifying this way salient areas.

A new bottom-up saliency model called *Graph-Based Visual Saliency* (GBVS from now on) was proposed by *Jonathan Harel et al.* in 2007 [13]. This algorithm consists in two simple main steps: First, features from the given image based on biological fixations are extracted, building several feature maps with them. Finally, these feature maps are combined and normalized forming the final saliency map. The model provided good results predicting human fixation on 749 variations of 108 natural images, achieving the 98% of the ROC area whereas [12] only achieves the 84%.

2.1.2. Dynamic saliency approaches

As it could be appreciated so far, significant progress has been done in terms of static saliency. However, it is clear that dealing with dynamic saliency or better said, saliency in videos would be even more challenging, opening new case studies for this field. It is important to point the fact that a video frame is observed for a fraction of second, while an image can be viewed for a long time. This fact and other considerations make video saliency estimation methods differ from image saliency methods, considering motion as an extension of static saliency. This way, dynamic saliency is defined, a concept that is gaining interest nowadays.

For instance, *Seo* and *Milanfar* [14] proposed a framework for static and space-time saliency detention (*SEO* model). This model was based on center-surrounding differences comparing a window centered on a pixel location with its neighboring windows. In order to include a dynamic measure, motion was considered for saliency detention turning the windows and their surrounding region into spatio-temporal cubes. In order to determine the salience of a point, features extracted from both, the centered window and from its surrounding area, were taken into account. This conditional probability of salience was estimated using a *Kernel Density Estimation (KDE).*

Another example in which static and dynamic features were included, is the saliency model proposed by *Culibrk et al.* [15]. This algorithm employs a multi-scale model in the form of a Gaussian pyramid as [12]. It also reduces the saliency values assigned to pixels that form static objects along the frames of the video, giving importance to objects in motion. The results conclude that saliency motion features improve prediction. In addition, the perception received by the viewer is also modified by blocking artifacts in salient areas and by the variance of temporal changes in non-salient parts.

On the other hand, *Mancas* model [16] only used dynamic features such as speed or direction, so no static features were added to this model. The algorithm is divided in three steps: first, a motion features extraction is made relying on Farneback's algorithm

for optical flow [17]. Once the features are discretized in 4 directions (north, west, east and south) and 5 speeds, a low-pass filter is applied to these channels, providing to the features a spatio-temporal description, which is the next step. Finally, the feature channels are fused providing a saliency index for each pixel.

After detailing all of these previous researches it is important to note that not everybody agrees that saliency map can perfectly describe a natural process such selective attention. For instance, *Rufin VanRullen* (2003) [18] argues against this classical view alleging that "visual processing tasks are clearly performed too fast for such a costly strategy to be employed". Moreover, it is explained that saliency can be implicitly represented through the ventral visual pathway, discarding any explicit saliency map. As conclusion, this research explains that object recognition influences directly to visual attention, being the object that dominates the perception the one whose features are most "salient".

2.2 Applications

The original application and motivation to develop the saliency map was focused on attention, in order to study which kind of stimulus and factors affect to it. More specifically, on covert attention which is the act of paying attention without moving the eyes, the contrary of overt one, in which the attention focus is shifted by an eyes movement. Despite this original idea, saliency map has been applied for other fields, for instance to study eye movements as the research studied by *Parkhurst et al.* in 2002 [19] in which the stimulus that guide the allocation of attention was deeply studied.

The saliency concept has been applied for data transmission as well. There are numerous of technical applications which make use of the saliency map in data processing, identifying which parts of the information should be prioritized in data treatment. Through this knowledge, communication systems dealing with video streams or images can be modified, improving the methods to generate, transmit and receive visual data.

For instance, Parkhurst and Niebur in 2002 [20] deal with techniques which are able to present the visual content in a display with several resolutions. In addition, most of the computational resources are located in those parts of the image that require more detail, regarding with the perception of the viewer. The principal advantage of this implementation is based on reducing the computational resources needed to deal with visual data.

Other clear example of the saliency utility in communication systems can be the model presented by *Courty* and *Marchand* in 2003 [21]. Based on the saliency features, visual perception was simulated for the development of a synthetic human character and for a video surveillance application. It is important to point that this research requires dealing with spatiotemporal information, so image motion analysis was added to the study.

Going further still, saliency map can deal with interactive 3D applications as the research presented by *Hillaire et al.* in 2010 [22]. In this paper, saliency map is chosen as the visual attention model to improve the accuracy of gaze tracking systems. The algorithm locates an uncertainty window around the gaze point defined by the viewer, basing the size of the window on the gaze accuracy of the same subject. After this, the algorithm tries to find another better gaze point contained in the window, improving the gaze tracking system. The method was demonstrated in two different contexts, a free exploration of a 3D environment; in which no task is demanded and a videogame based on gaze tracking involving a selection task.

Not all the researches related with this concept have been focused on salient parts specifically. For instance, *Su et al.* (2004) [23] studied ways to modify the natural behavior, trying to redirect the human attention to specific imposed regions. Image processing algorithms were used in order to impose saliency contrast in the image, altering the standard saliency map of the processed image.

2.3 Motivations and Proposal

We can observe that the application of saliency map has been deeply involved with eye-tracking and simulations regarding visual attention. However, this research has been motivated by other applications related with saliency, relying on video classification. The quality of a video and the way it is perceived by the viewer mostly depends on the aesthetics. In consequence, the perception of a still image or a video not only depends on the content, there are some objective data that might cause a subjective impression.

Despite this matter was explained in more detail in [2], this research is focused in the same concept of how objective data, as saliency in this case, can highly modify the subjective perception of a video. Relying on this affirmation, a classification model based on aesthetics visual features was developed in [2], demonstrating that this collection of features has an important impact in video perception. In order to achieve this result, a video database of 138 car commercial videos was defined, classifying the instances by the model implemented. 72.18% of the videos were correctly classified based on this aesthetics features.

This research is motivated by the extension of this commented experimentation, working with the same dataset of videos and adding new objective data to enforce his

affirmation. This objective data is only concerned with the salience and a new classification model will be developed, studying this way how saliency might vary the perception of the viewer. Better said, it will be studied which parts of the related images are prioritized by visual attention, and how the movement, shape, size and other characteristics of these areas affect to the perception of this collection of 138 videos extracted from YouTube.

2.3.1. Saliency based on image and video classification

Next, some previous classification models based on saliency are named and described. This might be helpful in order to confirm if saliency features are descriptive enough to implement a satisfactory classification system, which would motivates our implementation. Through the results retrieved by these models, we can describe a background for our purpose, expecting to obtain similar results.

A simple contribution of the saliency map can be demonstrated in [24], a research in which a monument recognition model was implemented. Having a dataset of images of well-known monuments, the goal was to classify properly new images of these monuments; which were taken from different angles or zooms, employing Scale Invariant Feature Transform (SIFT) or Speeded Up Robust Features (SURF). It was demonstrated that preprocessing the images with a saliency model such as GBVS [13], improved the results obtained, as well as the computational time needed in the matching process using SIFT and SURF.

In [25], a new saliency detection model was proposed. This model was not focused on obtaining the best descriptors set, selecting three specific descriptors: color, intensity and spatiotemporal orientation. This way, the saliency map is obtained and segmented in regions which are ordered by their saliency value. Through these regions, different features are extracted in order to create a global descriptor to use in the classification process, using a five-fold cross validation as classification procedure.

The dataset used for this project was formed by a collection of 924 clips of 7 different kinds of sports. In addition, different multi-class video classification algorithms were compared with the proposed one, which provides the best classification result.

Xiao Lv et al. [26] proposed a classification model based on the saliency as well, incorporating fuzzy reasoning rules. First, this model was proved in several datasets for image classification, improving the results retrieved by other previous coding methods that are compared. In relation to our domain; videos, also the model was used for an elevator surveillance video classifier. The classes were defined regarding to overload or violence detection in elevators and the classification was implemented by linear SVM. Half of the frames of each video were used as the training set, whereas the other half

was used as the testing, providing good results and demonstrating the effectiveness of the proposed model.

As it can be seen in these previous works, saliency does have an important impact in classification procedures in which different image or video classes are involved. Despite these models demonstrate the saliency importance when classifying, their results do not support any conjecture previously defined, i.e. their main goal is to classify new incoming images/videos as well as possible, based on the proposed model. Moreover they aim to improve the results obtained by previous methods.

On the other hand, our model not only implements a classifier but also, tries to verify if an initial assumption is correct through the results obtained. This supposition, as commented before, is based on the subjective perception of a video and how the saliency affects directly on it.

As conclusion, the motivation of this paper principally relies on an extension of experimentation treated in [2], in which saliency will be the main issue to work with. A classifier will be developed through a new model, which will study the saliency map of the frames belonging to the videos of the collection. More specifically, most-salient parts of the frames will be defined, focusing the study on them. In terms of saliency, these salient regions are known as *blob*. Thus, a new collection of features will be extracted from the blobs, using it for the training and the testing of the classifier implemented. Finally, the results obtained by the classifier will discuss the initial stablished assumption.

The results of the classification methods will verify the usefulness of saliency in the classification process and as consequence, will discuss the impact in subjective perception. In addition, a different background is defined for this proposal, where videos related with advertising; car commercials in this case, will be used.

Chapter 3

Implementation

In this chapter, the initial steps of the proposed computational model are explained. Thus, different techniques, decisions, as well as the implementations related with the saliency map and saliency features are detailed. First, the most influential saliency algorithms [12] and [13] are explained in more detail, as well as discussed and compared. Applying one of these algorithms to the database, the main process of saliency features extraction will be studied, detailing finally a collection of features used for our proposed model based on saliency.

3.1 Discussion of algorithms

In the state of the art of this paper we have presented several contributions concerning with saliency. Despite we are analyzing videos, and both static and dynamic saliency approaches have been mentioned, we are relying on static saliency solutions. Dynamic behavior will be incorporated in the selection of features, which will be explained further.

As previously commented, the saliency values assigned to the pixels are dependent of discontinuities in the algorithm presented in [12]. Since a binarization will be applied in our process, which will be explained later, fragmented regions can be obtained, being this algorithm a good option for our purpose. Also, [13] should be discussed since it provides good results concerning human attention, which is the basis of our research.

Below, a scheme of the process of a typical static saliency model is described; distinguish between three main steps, feature extraction, saliency measurement and normalization.

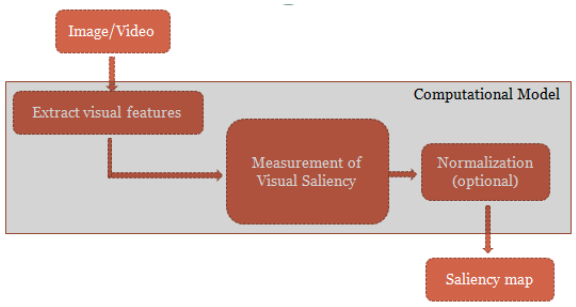

Figure 3.1: Static saliency models scheme

3.1.1 Itti, Koch and Niebur algorithm [12]

In the first phase of this model (Itti's from now on), a *linear filtering* process is performed, reducing the image to nine different scales using Gaussian pyramids. This is a common technique used in image processing, which makes escalated copies of the same image. In order to do so, the average of a pixel neighborhood related to the original image is calculated, building this way, all the new pixels of the escalated image. Thus, each performed iteration corresponds to a new escalated image as shown in the example in Figure 3.2

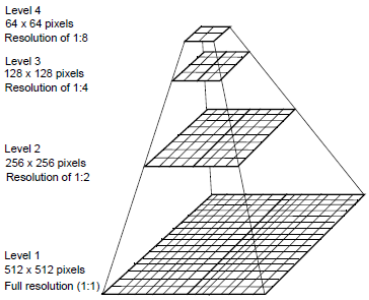

Figure 3.2: Gaussian pyramids algorithm

However, this model does not apply the algorithm directly. Three set of images are previously extracted from the main image, attending to its intensity, colors and orientation, working with these sets separately.

The first set of images is related to intensity. With *r*, *g* and *b* as the red, green and blue channels of the image in question, the intensity image *I is* calculated as shown in the 3.1 equation

$$I = \frac{r+g+b}{3} \tag{3.1}$$

Once image *I* has been calculated, it is converted to several scales using Gaussian pyramids algorithm previously commented. This way, we will obtain *I(σ)*, where $\sigma \in [0..8]$ is the scale

The second set is referred to color. Four broadly-tuned color channels are created, normalized by intensity and finally, reduced to zero in pixels where the intensity value does not reach at least the 1/10 of its maximum over the entire image. Therefore, the color channels are described as follows:

$$R = \frac{r-(g+b)}{2}\ G = \ \ B = \frac{b-(r+g)}{2}\ \ Y = \frac{(r+g)}{2} - \frac{|r-g|}{2} - b \tag{3.2}$$

Four Gaussian pyramids are created for these four color channels *R(σ)*, *G(σ)*, *B(σ)* and *Y(σ)*, with the same scales as done with image *I*.

Finally, the same procedure is executed for orientation, obtaining $O(\sigma,\theta)$, where $\sigma \in [0..8]$ is the scale and $\theta \in \{0°, 45°, 90°, 135°\}$.

Once these sets have been extracted, "center-surround" operation is implemented in order to detect regions that stand out from their surroundings. In order to understand this operation, it is important to know that visual neurons are more sensitive in the center of the visual space, whereas the surrounded region inhibits the neural response. In addition, such architecture can be applied also for local spatial discontinuities, detecting locations which locally stand out from their surround. This way, "center-surround" operation is implemented in the model, differencing between a "center" scale and a "surrounding" scale, which yield the feature maps (42 in total).

Considering the center as a pixel at scale $c \in \{2,3,4\}$, the surround as the respective pixel at scale $s = c + \delta$ (where $\delta \in \{3,4\}$) and finally denoting the "center-surround" operation with the symbol $\ominus$, we can retrieve all the feature maps, attending to the formulas described below.

$$I(c,s) = |I(c) \ominus I(s)| \tag{3.3}$$

$$RG(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \tag{3.4}$$

$$BY(c,s) = |(B(c) - Y(c)) \ominus Y(s) - B(s))| \tag{3.5}$$

$$O(c,s,\theta) = |O(c,\theta) \ominus O(s,\theta)| \tag{3.6}$$

These feature maps are combined in three separate "conspicuity maps", $\bar{I}$ for intensity, $\bar{C}$ for color and $\bar{O}$ for orientation through across-scale addition, "$\oplus$". After this, the conspicuity maps are normalized and summed, retrieving finally the saliency map. The whole process is shown in the following flow diagram:

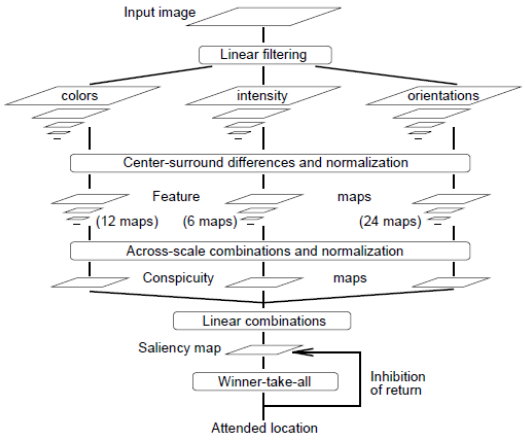

Figure 3.3: Itti and Koch saliency model

3.1.2 GBVS algorithm [13]

This model based on bottom-up mechanisms was proposed by *Jonathan Harel*, *Christof Koch* and *Pietro Perona* [13]. Given an image *I*, this study tried to estimate what part of the image would be the most significant by a human being according its fixation. This model is computed following three main steps; as other saliency models, which were previously named.

1) **Extraction**: We need to highlight significant locations where the image is informative according to the criteria of human fixation. In order to do so, feature maps are computed by filtering based on biological mechanisms of vision. For instance, applying a linear filtering followed by some elementary nonlinearity [27], which is out of scope of this research. This linear filtering is made in order to reduce the noise of the image, enhance the image and get it ready for the feature extraction.

2) **Activation:** Given a feature map $M : [n]^2 \rightarrow \mathbb{R}$, which has been obtained from step 1), the goal is to create an activation map $A : [n]^2 \rightarrow \mathbb{R}$. The pixels $(i,j) \in [n]^2$ belonging to feature map *M*; which are unusual related to their neighborhood, will be treated as high values for the activation map *A*. There are several approaches to define these "unusual" locations. For instance, we can implement either the "center-surround" operation or a histogram of *M(i,j)* computed in the region *(i,j)*, treating it as a probability distribution.

On the other hand, we can define these locations using a Markovian approach. First of all, the *dissimilarity* function (3.7) is introduced in order to measure differences between pixels of the feature map *M*. Basically, it is based on the distance between the pixels, measured on a logarithmic scale. This would be the expression in order to compare the pixels *M(i,j)* and *M(p,q)*:

$$d((i,j)||(p,q)) \triangleq \left|\log\frac{M(i,j)}{M(p,q)}\right| \tag{3.7}$$

The map is now conceived as a fully-connected directed graph G_A, where every pixel with two indices $(i,j) \in [n]^2$ is compared with all other $n - 1$ nodes or pixels. This way, every edge defined from node *(i,j)* to node *(p,q)* will be assigned a weight:

$$\omega_1\big((i,j),(p,q)\big) \triangleq d((i,j)\big|\big|(p,q)\big)\, F(i-p, j-q) \text{ , where}$$

$F(a,b) \triangleq exp\left(-\frac{a^2+b^2}{2\sigma^2}\right)$, where σ is one tenth to one fifth of the map width (3.7)

As it can be seen, this weight is directly proportional to the dissimilarity and to the closeness of the pixels in comparison. This way, if we are comparing close pixels with high dissimilarities in their values based on the feature map, a high weight will be

assigned. On the other hand, a lower weight will be assigned for distant pixels with similar values.

Now, we can trace a Markov chain on G_A. This Markov model is based on a stochastic process defined by states or events and a probability distribution for each state. The probability of each state only depends on the previous one and not on a sequence of events. Therefore, each node will be labelled as a state, whereas edge weights as transition probabilities. Furthermore, in order to normalize G_A, the highest edge weight of each pixel will be rescaled to unity.

This equilibrium distribution will be the basis to build the activation map A, and it can be considered as an "organic" solution since each node works independently as neurons do. The result will be the equilibrium state retrieved when compiling the inputs of all the states/nodes of the chain.

Due this communication, neurons are able to include fast decisions about which areas of a scene require more attention by the viewer and, as consequence, an additional processing is required as well. For this method, we have the same case, since it exposes connected (via F) regions of dissimilarity (via w) computed in a parallel way.

Furthermore, as neuron networks do, computations can be carried out independently, following a synchronous and simultaneously way by each node. Thus, for each moment, each node sums incoming mass and then, the node makes partitions of this mass passing them to its neighbors. This partition is made in relation to their edge weights, being likely to accumulate mass in close states with high edge weights

3) **Normalization/Combination:** The main goal for this step is to concentrate the mass on activation maps. If the mass is not concentrated prior to the combination step, then the master map obtained may be uniform and its information may be useless.

To solve this problem, normalization is required. In order to achieve it, another Markovian algorithm is used. Given an activation map $A : [n]^2 \rightarrow \mathbb{R}$ a graph G_N is built. This graph has n^2 nodes representing regions and edges with weight values. The weight values are proportional to the activation map values and the relative distance between the nodes as defined below:

$$\omega_2\big((i,j),(p,q)\big) \triangleq A(p,q)F(i-p,j-q) \tag{3.8}$$

Again, after normalizing each highest weight for each node to unity, it can be verified how mass stays preferentially in those nodes with high activation, so an equilibrium

distribution can be computed. This normalization seems to generate better results in comparison with other standard approaches, such as “DoG” or “NL”.

Some researchers have contributed to these steps. For example, in order to create the map (step 2) and to normalize it (step 3), Bruce [28] and others [29] are good contributions respectively. Both hypothesized that this map was formed by fundamental quantities as “self-information” or “surprise”. [28] proposed an additive function for feature maps in such a way to get the activation map.

3.2 Main process

In this work, we try to verify how important the saliency is for aesthetics assessment studies and for the subjective perception of video viewers. In order to do so, a new model will be developed, relying on the same collection of videos that were used in [2].

First, the saliency map extraction process is applied. It is important to point that we should take into account homogeneous images when working with saliency algorithms. For instance, when dealing with entire black images, unusual pixels cannot be retrieved since all of them are the same, making the feature maps extraction impossible to achieve. To face this problem and before starting with the saliency map extraction process, it is necessary to verify what kind of image we are dealing with.

3.2.1. Saliency map extraction process

Starting from this point, the related frames of the videos are extracted as done in [2]. Once the frames are ready, we are able to stablish an algorithm to get a saliency map for each frame. Finally, GBVS algorithm was chosen to achieve this duty.

This decision was taken due to the accuracy of the algorithm, being this one more predictive in comparison with Itti’s. In order to explain this, we should take into account two important factors for saliency algorithms. The first one is the width of the final blur obtained when calculating the probability distribution or the Markov chain, getting the master map. The other one is the center bias of this master map.

These two factors can be visualized in the image described below, where both GBVS and Itti’s algorithms were applied in an example image.

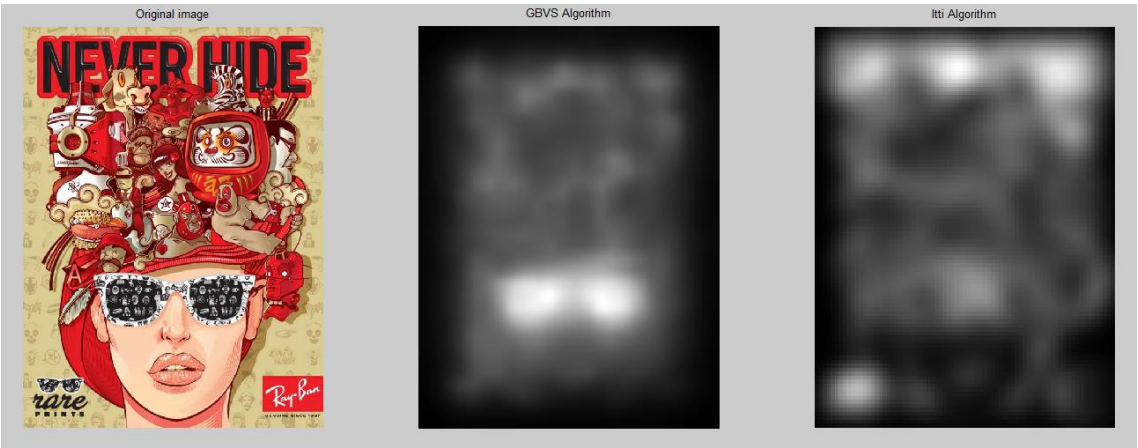

Figure 3.4: GBVS and Itti's algorithms comparison

As it can be seen, GBVS algorithm retrieves a more accurate result of saliency, providing a clearer and more compact distribution with less deviation.

Furthermore, this comparison took place previously in other studies, verifying that better results can be obtained by GBVS algorithm. For instance and relying on the affirmation studied in [10], an interesting study about the fixation of the human eye on faces was carried out in 2007 [30]. Having a database of more than 200 images with human faces, they obtained a 0.841 mean ROC using GBVS algorithm, versus 0.812 mean ROC for Itti's. This means that almost the 85% of the images using GBVS algorithm gave human faces as result of the most saliency part of the treated picture.

A simplified version of Itti's algorithm was also used for a lower computation cost. It is expected to provide a faster and a more accurate result than Itti's algorithm.

An example of the same image treated before is shown below.

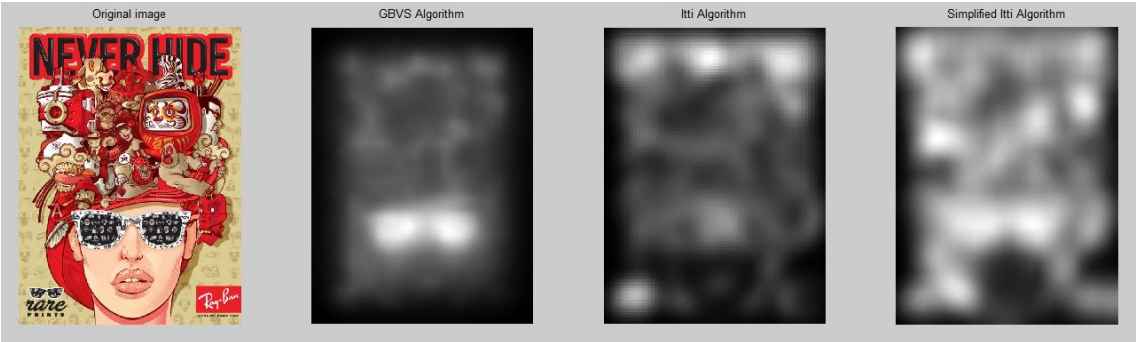

Figure 3.5: GBVS, Itti and Itti simplified algorithms comparison

This simplified version is conceived as a more faithful reproduction of the algorithm presented in [12], including the same center surround operation. This affirmation can be the consequence of the clear differences in the results provided by both, the simplified

version and the original one. Although, GBVS still provides a more compact solution. In addition, "GBVS promotes higher saliency values in the center of the image plane" as concluded in [13], being this center bias favorable to predict human attention.

For the choice of the algorithm to apply, it should be appropriated to discuss between computational cost and quality obtained. For our research, a really high cost is needed since we are dealing with videos and, as consequence, with a big amount of frames. As conclusion, a saliency map should be extracted for all the frames, which involves a really demanding process.

An optimization of the strategy could be a good choice for this matter. For instance, we could apply video compression due to the high temporal redundancy of the videos, reducing this way the amount of frames to work with.

However, the lessening of this cost is out of scope of the aimed goal, since this research is directly focused on other matter, an experiment in which the main hypothesis is related with the saliency and its influence in videos perception.

Due these reasons, GBVS algorithm was chosen as commented before, obtaining a better quality. Although, a higher computational cost is needed, which is not especially relevant for our purpose.

3.2.2. Segmentation process

Once the map is obtained, a segmentation process should be done. The main goal of this process is to discard no important information, analyzing only the information related to our proposal, compact and solid salient regions.

Since the map is formed by different levels of saliency, the first attempt to classify these levels was to make a segmentation process into different groups, discussing about how many levels or groups should be retrieved.

This way, each image should be split in different regions or sets. These regions are defined by pixels that share similar characteristics. This concept is known as *clustering*, in which every region retrieved by the process should be defined by a *centroid*.

In order to accomplish the clustering process, *k-means* algorithm was proved. The main goal of this algorithm is based on partitioning n observations, into k groups or classes called *clusters*. This way, each pixel would belong to the group or *cluster* with the nearest mean of the characteristic in question.

Thus, given a set of observations values x, *k-means* clustering aims to classify these observations into k groups, minimizing the sum of squares as follows:

$$\arg min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

,where μ_i is the mean of each defined group, S_i (3.9)

In this case, the observations are referred to the brightness of each pixel that forms the saliency map in question. Since we are retrieving a map with different levels in gray scale, the image is going to be labeled in k groups, classifying each pixel to the group of pixels with the nearest mean referred to the brightness.

This kind of algorithm can be perfectly implemented using Matlab, defining a brightness value for each pixel and identifying its most appropriate group according to k-means. Using the same image example, the results retrieved using six, three and two groups are shown below.

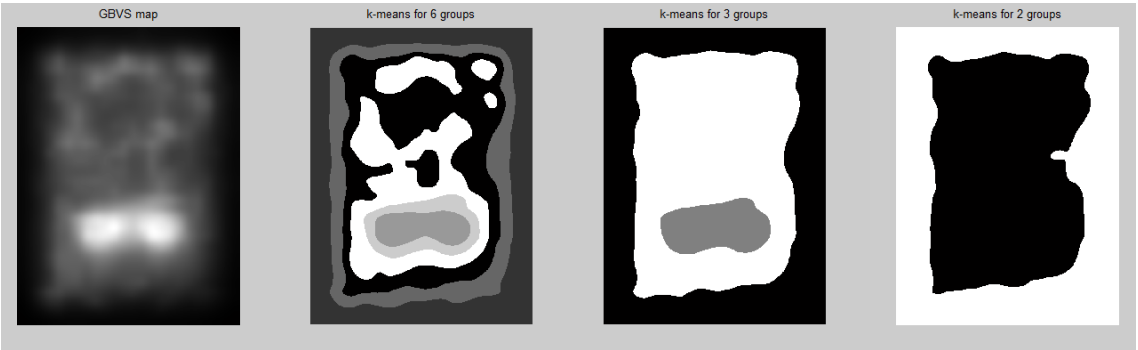

Figure 3.6: K-means performance comparison

Observing the obtained GBVS map, it would be appropriate to say that the most salient region is defined by the group of pixels which form the sunglasses. Thus, high brightness values have been assigned for this group of pixels.

Checking the results, the best solution; taking into account this affirmation, should be the one retrieved when using six groups, since the sunglasses are perfectly defined. This way, the group of pixels that defines the glasses would be analyzed and the other ones discarded, making this model totally valid for this specific training set. However, this solution might not be the appropriated one when using other pictures as new training data, because some important information needed for the analysis might be lost. We would be facing an *overfitting* problem.

Just the opposite is happening when defining two groups. In this case, a higher differentiation between pixels brightness is necessary to identify the most salient regions properly. Thus, this model does not fit properly with this specific training set.

As conclusion, maybe it is correct to think to define three groups is the optimized solution for this case, but we do not know what kind of frames extracted from a big amount of car advertisements we are going to face with. Due this, binarization is a better strategy to apply, providing this way a more simple, effective and general solution in order to segment the saliency map. Furthermore, this strategy provides enough information for our goal, since we simply aim at relying on particularly salient regions of the image, the blobs.

To perform this binarization, a threshold has been stablished at 0.75 taking into account that all the pixel brightness values in gray scale are defined in the scale (0,1). This value has been chosen as the most robust solution, and tested with different frames extracted from the video collection. This way, all the pixels that overpass the established threshold, will be applied 1 as brightness value, and on the other hand, those pixels that do not overpass the threshold, will be applied a 0 in their brightness value, making two clear labels for each image. Hence, a simplified version of the saliency map focused on the blobs is obtained.

When applying *k-means* algorithm, all the pixels related to their respective group or *cluster* are together, thus a connection exists between them. This might not be the same case as when applying a threshold, so in order to include all the information retrieved with this algorithm, the features that are going to be extracted will be directly studied separately for each blob.

After applying the threshold previously commented, we will obtain a binarization of the map image in which the blobs are located. Identifying these blobs is an important matter in order to extract their features separately. We will deal with it through *bwconncomp* Matlab function, which finds and identifies all the connected components in a binary image.

After this and working with the Matlab image processing toolbox, *regionprops* function was applied, extracting different properties about the shape and movement of the blobs in order to define the main frame features. This function takes into account the number of connected components provided by *bwconncomp,* and then returns a struct array containing a struct for each object in the image. All the simplified saliency maps are obtained in this way.

Below, an illustration of this commented process is described. Note this process is repeated for each video, providing all the frame features for the video in question

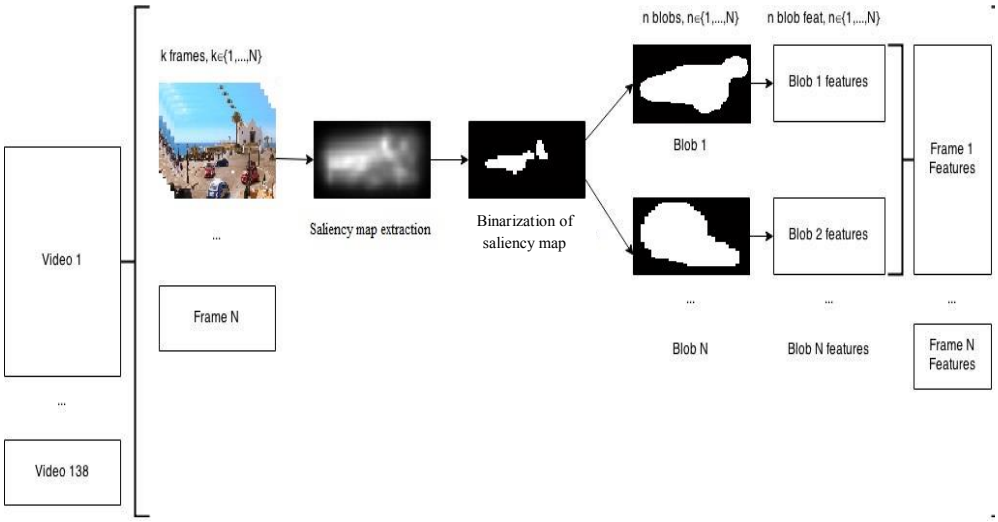

Figure 3.7: Total process scheme

3.3. Collection of Features

At this point, we are exclusively dealing with the features selection to conform our model. Features related with pixels values are not taken into account, since the pixels that form the blobs only have one intensity value, 1 due the binarization. Relying on regionprops function the collection of features is obtained. Although, some features provided by the function referred to shape were discarded, since they implied a modification of the blobs (which is out of the scope of this research) or because they provide useless information for saliency study field. Furthermore, some additional features were included in the collection, *Complexity, Speed* and *Acceleration* which will be explained further.

These selected features will contribute to generate the main video features through means and standard deviation formulas, being important to clarify that they are low-level features. Low-level features are those ones that are directly extracted from the image that is being computed. Thus, no human subjective decision affects the generation and extraction of these features, which would be part of high-level features analysis.

Since the base of the main objective of this work is according to the importance of saliency in subject perception of videos, all the features have been extracted from all the frames of the videos. Then, a temporal dimension meaning has been provided to the features processing mean and standard deviation calculations for each video.

The whole set of used features for the project is described below. It is important to point that binarization typically produces several fragmented salient regions which can be sorted upon their corresponding area. This way, two types of descriptors can be extracted from a frame due to binarization. The first type is referred to the biggest blob of the frame (labeled as *biggest*) whereas the second type is referred to all blobs or overall saliency, defining a global and unique measurement of the descriptor in question (labeled as *overall*). *Number of blobs* feature does not have *biggest* and *overall* concepts defined though. On the other hand, *Speed* and *Acceleration* frame features are only defined for the biggest blob since the overall one cannot be defined in a logical way

Despite the solution to design our model has a static behavior (i.e. dynamic measurements are not added to the features collection), the features collection selected pretends to model dynamic saliency aspects as well as its evolution along the whole video. For instance, motion related features are included in the collection.

Number of blobs

It is referred to the amount of saliency blobs retrieved within an image after GBVS algorithm and binarization of the map are applied. This way, it calculates the number of groups containing connected pixels with 1 as brightness value. It could determine how many attention spots a human being would be focused on. In order to explain this feature visually, a frame extracted from one of the videos of the collection is described below. In this case, the unique blob found is highlighted whereas the rest of the image has been obscured.

Figure 3.8: Number of blobs example

In Table 3.1, we can observe different calculations of this feature for our experimentation. It is important to point that minimum value of the number of blobs mean is lower than 1, indicating that no blobs were found in some frames. This problem was commented in the previous chapter and has been faced with the development of a black image detector. This way, if the algorithm is treating with a really dark image or an entire black one, no saliency map is obtained, giving to the features a default value.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_1	meanNumOfBlobs	0	∞	0.321	3.109	1.699	0.478
f_2	stdNumOfBlobs	0	∞	0.363	1.798	1.144	0.228

Table 3.1: **Number of blobs** features results

3.3.1 Morphology related features

Area

It is referred to the number of pixels that define the entire blob. This way, it could determine how large the blobs are and how they change their size along the entire video. Thus, a low value of the standard calculations could be an indicative of a video with few changes in the size of its blobs, and as consequence, these sizes might be related to the same blobs (same saliency areas).

This feature has been normalized to unity, being the unity the area of the whole image, H x W (Height multiplied by width). As it can be observed, the values retrieved are really far from 1, meaning that the entire image should be never treated as a unique salient region.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_3	meanBiggestArea	≈0	1	0.001	0.039	0.015	0.006
f_4	stdBiggestArea	≈0	1	0.002	0.035	0.013	0.006
f_5	meanOverallArea	≈0	1	0.001	0.044	0.019	0.008
f_6	stdOverallArea	≈0	1	0.002	0.037	0.013	0.006

Table 3.2: **Area** features results

Perimeter

This feature calculates the number of pixels which defines the boundary that encloses the blob. Since the blobs are continuous regions, a boundary can be defined for each blob with no exceptions.

It has been normalized to unity, being the unity the perimeter of the whole frame, i.e. $2H + 2W$

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_7	meanBiggestPerimeter	≈0	1	0.014	0.212	0.113	0.035
f_8	stdBiggestPerimeter	≈0	1	0.023	0.141	0.074	0.021
f_9	meanOverallPerimeter	≈0	1	0.019	0.348	0.165	0.054
f_{10}	stdOverallPerimeter	≈0	1	0.025	0.189	0.117	0.029

Table 3.3: **Perimeter** features results

Complexity

It calculates how complex the blobs are. In order to retrieve this result, the ratio between the perimeter and the area belonging to the same blob has been used. This way, a blob with a low perimeter value and high area value would be more compact, less spread and therefore, with fewer complexes (for instance, a circle).

On the other hand, a blob with high perimeter value and low area value would be more spread, thinner and with a more complex shape.

Theoretically, the minimum value would be obtained if the perimeter of the blob is 0. Since this value is not logical, the theoretical minimum value for complexity would be approximate to 0, ≈0.

On the other hand, the maximum value would be obtained if the area of the blob is 0. For the same reason, the maximum value for complexity would be ≈∞

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{11}	meanBiggestComplexity	≈0	≈∞	1.17	12.978	8.14	1.968
f_{12}	stdBiggestComplexity	≈0	≈∞	0.023	7.948	3.787	1.242
f_{13}	meanOverallComplexity	≈0	≈∞	1.304	13.6	8.873	2.115
f_{14}	stdOverallComplexity	≈0	≈∞	1.241	8.052	4.073	1.301

Table 3.4: **Complexity** features results

Solidity

This feature is calculated using the ratio between the blob area and the area of the smallest convex polygon that can contain the blob. This way, if the blob fits perfectly in the smallest convex polygon, the value 1 should be retrieved, being a 0 value impossible to retrieve due to the existence of blob area.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{15}	meanBiggestSolidity	≈0	1	0.113	0.961	0.78	0.155
f_{16}	stdBiggestSolidity	≈0	1	0.059	0.487	0.282	0.123
f_{17}	meanOverallSolidity	≈0	1	0.114	0.967	0.796	0.159
f_{18}	stdOverallSolidity	≈0	1	0.042	0.486	0.278	0.132

Table 3.5: **Solidity** features results

The basis of these first commented features relies on the area of the blobs. In order to understand them in a visual way, an example of a frame is described below, showing blobs with different areas varying the value of these features.

Figure 3.9: Blobs areas example

Eccentricity

This feature measures of how much the blobs deviate from being totally circular. This way, degenerate values could be retrieved, having a circle if the eccentricity is 0 and a line segment if the eccentricity is 1.

This feature could determine the shape of the blob in question and helps to know how its shape changes along the video.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{19}	meanBiggestEccentricity	0	1	0.098	0.826	0.647	0.137
f_{20}	stdBiggestEccentricity	0	1	0.1	0.42	0.272	0.082
f_{21}	meanOverallEccentricity	0	1	0.088	0.793	0.631	0.133
f_{22}	stdOverallEccentricity	0	1	0.071	0.41	0.252	0.087

Table 3.6: **Eccentricity** features results

Major axis length

This feature describes the length; in pixels, of the major axis of the ellipse described with the same normalized second moments as the blob. This way, a high value could be related to an extended blob, whereas a low value is an indicative of a more compact blob.

It has been normalized to unity, being the unity the maximum line segment that could be described within the frame, i.e. the diagonal of the frame, $\sqrt{H^2 + W^2}$.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{23}	meanBiggestMajorAxis	≈0	1	0.015	0.205	0.114	0.035
f_{24}	stdBiggestMajorAxis	≈0	1	0.021	0.151	0.077	0.021
f_{25}	meanOverallMajorAxis	≈0	1	0.009	0.149	0.089	0.026
f_{26}	stdOverallMajorAxis	≈0	1	0.02	0.124	0.061	0.019

Table 3.7: **Major Axis** features results

Minor axis length

This feature describes the length; in pixels, of the minor axis of the ellipse described with the same normalized second moments as the blob.

It has been normalized to unity as done with the major axis length feature.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{27}	meanBiggestMinorAxis	≈0	1	0.007	0.107	0.06	0.018
f_{28}	stdBiggestMinorAxis	≈0	1	0.014	0.078	0.033	0.009
f_{29}	meanOverallMinorAxis	≈0	1	0.005	0.086	0.048	0.013
f_{30}	stdOverallMinorAxis	≈0	1	0.012	0.066	0.027	0.008

Table 3.8: **Minor Axis** features results

Circularity

It calculates how circular the shape of the blob is. The value has been retrieved using the last two described features, making the ratio between the minor axis length and major axis length of the ellipse.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{31}	meanBiggestCircularity	≈0	1	0.061	0.695	0.5	0.107
f_{32}	stdBiggestCircularity	≈0	1	0.111	0.418	0.25	0.055
f_{33}	meanOverallCircularity	≈0	1	0.068	0.667	0.505	0.106
f_{34}	stdOverallCircularity	≈0	1	0.083	0.417	0.234	0.062

Table 3.9: **Circularity** features results

These last features are directly referred to a comparison of the blob with a circle. In order to illustrate different cases concerning with this comparison, two examples in the same frame are described below, showing opposite cases. A circular blob is described in the car headlight, whereas an extended one highlights the license plate.

Figure 3.10: Circularity of blobs example

Orientation

This calculation returns a scalar that specifies the angle between the major axis of the ellipse with the same second moments as the blob, and the x-axis of the frame.

Since this value has not been normalized and is ranging from -90 to 90, the theoretical maximum value that the standard deviation could take would be calculated as follows:

The maximum standard deviation would be obtained from the video with the biggest amount of frames, X. Thus, having a 90 degrees orientation (*biggest* or *overall*) in X/2

frames and -90 degrees orientation (*biggest* or *overall*) in the other half, the scenario with highest deviation could be obtained. The largest video is formed by 3141 frames, providing this way the commented calculations.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{35}	meanBiggestOrientation	-90	90	-16.3	46.385	0.885	7.832
f_{36}	stdBiggestOrientation	0	90.01	4.598	69.922	44.621	9.367
f_{37}	meanOverallOrientation	-90	90	-19.7	37.229	0.438	6.754
f_{38}	stdOverallOrientation	0	90.01	5.712	56.928	35.359	7.618

Table 3.10: **Orientation** features results

Figure 3.11: Orientation of blobs example

In Figure 3.11, an example of different orientations for the blobs in the same frame is described.

Extent

This feature calculates the ratio between the pixels contained in the blob and the pixels that define a bounding box, being the bounding box the smallest rectangle that can be described containing the whole blob.

This way, the higher the value retrieved, the more compact the shape of the blob, having a rectangle when the value is 1. On the other hand, a 0 value cannot be obtained since it would be extracted from a blob area with value 0.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{39}	meanBiggestExtent	≈0	1	0.082	0.709	0.563	0.112
f_{40}	stdBiggestExtent	≈0	1	0.07	0.38	0.227	0.078
f_{41}	meanOverallExtent	≈0	1	0.085	0.727	0.586	0.116
f_{42}	stdOverallExtent	≈0	1	0.049	0.381	0.22	0.089

Table 3.11: **Extent** features results

This feature can be illustrated in Figure 3.11 as well.

3.3.2 Location related features

Centroids

This feature calculates the coordinates of the blobs. This way, it could determine the position of the blobs in reference to the whole frame. Low values in the standard calculations can be an indicative of a video in which the blobs are not changeable in their position. That means that saliency area is almost static, and as consequence, the perception of the video could be more comfortable for the viewer.

These coordinates have been normalized to unity, being the unity W for x-axis coordinates and H for y-axis coordinates.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{43}	meanBiggestCentroid_x	0	1	0.387	0.641	0.504	0.037
f_{44}	meanBiggestCentroid_y	0	1	0.349	0.629	0.497	0.032
f_{45}	stdBiggestCentroid_x	0	1	0.044	0.216	0.144	0.029
f_{46}	stdBiggestCentroid_y	0	1	0.032	0.179	0.101	0.025
f_{47}	meanOverallCentroid_x	0	1	0.417	0.636	0.504	0.031
f_{48}	meanOverallCentroid_y	0	1	0.354	0.623	0.497	0.030
f_{49}	stdOverallCentroid_x	0	1	0.036	0.181	0.115	0.027
f_{50}	stdOverallCentroid_y	0	1	0.027	0.165	0.088	0.023

Table 3.12: **Centroids** features results

Extrema

The feature calculates the coordinates of each extrema points in the blob in question. These extrema points are detailed in the figure below:

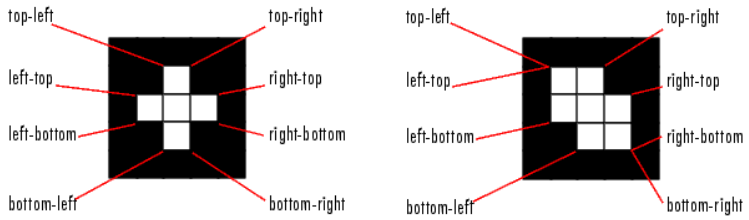

Figure 3.12: Extrema feature coordinates

The coordinates of adjacent extrema points could be the same, as some cases detailed in the table below, giving us an idea of how regular the blob is.

The collection of coordinates of this feature has been normalized to unity, being the unity W for x-axis coordinates and H for y-axis coordinates.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std
f_{51}	meanBiggestExtremaTopLeft_x	0	1	0.387	0.64	0.499	0.037
f_{52}	meanBiggestExtremaTopRight_x	0	1	0.395	0.644	0.51	0.037
f_{53}	meanBiggestExtremaRightTop_x	0	1	0.43	0.679	0.556	0.039
f_{54}	meanBiggestExtremaRightBottom_x	0	1	0.43	0.679	0.556	0.039
f_{55}	meanBiggestExtremaBottomRight_x	0	1	0.383	0.645	0.51	0.038
f_{56}	meanBiggestExtremaBottomLeft_x	0	1	0.372	0.638	0.499	0.038
f_{57}	meanBiggestExtremaLeftBottom_x	0	1	0.344	0.631	0.452	0.043
f_{58}	meanBiggestExtremaLeftTop_x	0	1	0.344	0.631	0.452	0.043
f_{59}	meanBiggestExtremaTopLeft_y	0	1	0.285	0.615	0.42	0.042
f_{60}	meanBiggestExtremaTopRight_y	0	1	0.285	0.615	0.42	0.042
f_{61}	meanBiggestExtremaRightTop_y	0	1	0.346	0.626	0.489	0.034
f_{62}	meanBiggestExtremaRightBottom_y	0	1	0.36	0.636	0.506	0.033
f_{63}	meanBiggestExtremaBottomRight_y	0	1	0.408	0.656	0.574	0.037
f_{64}	meanBiggestExtremaBottomLeft_y	0	1	0.408	0.656	0.574	0.037
f_{65}	meanBiggestExtremaLeftBottom_y	0	1	0.351	0.634	0.507	0.032
f_{66}	meanBiggestExtremaLeftTop_y	0	1	0.34	0.628	0.49	0.033
f_{67}	stdBiggestExtremaTopLeft_x	0	1	0.052	0.215	0.147	0.029
f_{68}	stdBiggestExtremaTopRight_x	0	1	0.053	0.217	0.147	0.029
f_{69}	stdBiggestExtremaRightTop_x	0	1	0.048	0.223	0.15	0.029
f_{70}	stdBiggestExtremaRightBottom_x	0	1	0.048	0.223	0.15	0.029
f_{71}	stdBiggestExtremaBottomRight_x	0	1	0.054	0.216	0.148	0.029
f_{72}	stdBiggestExtremaBottomLeft_x	0	1	0.054	0.216	0.148	0.029
f_{73}	stdBiggestExtremaLeftBottom_x	0	1	0.061	0.213	0.15	0.029
f_{74}	stdBiggestExtremaLeftTop_x	0	1	0.061	0.213	0.15	0.029
f_{75}	stdBiggestExtremaTopLeft_y	0	1	0.039	0.192	0.116	0.023
f_{76}	stdBiggestExtremaTopRight_y	0	1	0.039	0.192	0.116	0.023
f_{77}	stdBiggestExtremaRightTop_y	0	1	0.032	0.181	0.111	0.024

f_{78}	stdBiggestExtremaRightBottom_y	0	1	0.034	0.182	0.111	0.025
f_{79}	stdBiggestExtremaBottomRight_y	0	1	0.035	0.199	0.114	0.022
f_{80}	stdBiggestExtremaBottomLeft_y	0	1	0.035	0.199	0.114	0.022
f_{81}	stdBiggestExtremaLeftBottom_y	0	1	0.04	0.184	0.11	0.025
f_{82}	stdBiggestExtremaLeftTop_y	0	1	0.039	0.183	0.11	0.024
f_{83}	meanOverallExtremaTopLeft_x	0	1	0.417	0.635	0.499	0.031
f_{84}	meanOverallExtremaTopRight_x	0	1	0.424	0.639	0.509	0.031
f_{85}	meanOverallExtremaRightTop_x	0	1	0.452	0.644	0.545	0.032
f_{86}	meanOverallExtremaRightBottom_x	0	1	0.452	0.644	0.545	0.032
f_{87}	meanOverallExtremaBottomRight_x	0	1	0.416	0.64	0.509	0.031
f_{88}	meanOverallExtremaBottomLeft_x	0	1	0.407	0.634	0.499	0.031
f_{89}	meanOverallExtremaLeftBottom_x	0	1	0.382	0.627	0.463	0.035
f_{90}	meanOverallExtremaLeftTop_x	0	1	0.382	0.627	0.463	0.035
f_{91}	meanOverallExtremaTopLeft_y	0	1	0.291	0.61	0.436	0.037
f_{92}	meanOverallExtremaTopRight_y	0	1	0.291	0.61	0.436	0.037
f_{93}	meanOverallExtremaRightTop_y	0	1	0.35	0.62	0.49	0.031
f_{94}	meanOverallExtremaRightBottom_y	0	1	0.364	0.629	0.506	0.031
f_{95}	meanOverallExtremaBottomRight_y	0	1	0.411	0.636	0.558	0.033
f_{96}	meanOverallExtremaBottomLeft_y	0	1	0.411	0.636	0.558	0.033
f_{97}	meanOverallExtremaLeftBottom_y	0	1	0.356	0.628	0.506	0.031
f_{98}	meanOverallExtremaLeftTop_y	0	1	0.345	0.622	0.49	0.031
f_{99}	stdOverallExtremaTopLeft_x	0	1	0.036	0.183	0.117	0.026
f_{100}	stdOverallExtremaTopRight_x	0	1	0.034	0.184	0.117	0.026
f_{101}	stdOverallExtremaRightTop_x	0	1	0.032	0.191	0.12	0.025
f_{102}	stdOverallExtremaRightBottom_x	0	1	0.032	0.191	0.12	0.025
f_{103}	stdOverallExtremaBottomRight_x	0	1	0.037	0.185	0.117	0.026
f_{104}	stdOverallExtremaBottomLeft_x	0	1	0.038	0.186	0.117	0.026
f_{105}	stdOverallExtremaLeftBottom_x	0	1	0.043	0.189	0.12	0.026
f_{106}	stdOverallExtremaLeftTop_x	0	1	0.043	0.189	0.12	0.026
f_{107}	stdOverallExtremaTopLeft_y	0	1	0.039	0.15	0.099	0.02
f_{108}	stdOverallExtremaTopRight_y	0	1	0.039	0.15	0.099	0.02
f_{109}	stdOverallExtremaRightTop_y	0	1	0.026	0.161	0.094	0.022
f_{110}	stdOverallExtremaRightBottom_y	0	1	0.029	0.173	0.094	0.022
f_{111}	stdOverallExtremaBottomRight_y	0	1	0.044	0.182	0.098	0.022
f_{112}	stdOverallExtremaBottomLeft_y	0	1	0.044	0.182	0.098	0.022
f_{113}	stdOverallExtremaLeftBottom_y	0	1	0.036	0.17	0.093	0.023
f_{114}	stdOverallExtremaLeftTop_y	0	1	0.036	0.163	0.093	0.022

Table 3.13: **Extrema** features results

As it can be seen in the results, standard deviation is really low for most of the cases and both overall and biggest values have centric coordinates. This way, we could say that the blobs are mainly located in the center of the frames for those cases, being their movement not significant

Note that some corners of the blobs defined by values such as *TopRight* and *RightTop* have the same value, understanding that the blobs are clearly defined for those cases.

The conclusion stablished in [13] about central locations of the blobs, might be the reason of this behavior. However due to inherent saliency of some elements as human

faces as demonstrated in [10], other cases can be obtained in which the blobs are located in the periphery. This differentiation is detailed in the figure below.

Figure 3.13: Coordinates of blobs depending on human faces example

3.3.3 Motion related features

Speed

The feature is calculated through a comparison between the coordinates of the same blob (only the biggest for this case) in two consecutive frames.

Biggest blob coordinates of both frames are subtracted in pairs (x-axis and y-axis). It is important to point that the direction the blob is moving should be taken into account in order to obtain real and positive values. For instance, if the blob has moved backwards in reference to x-axis, we will obtain a negative value, which does not have any sense for this feature.

Since the coordinates have been already normalized, this feature is normalized to unity too, being the unity W for x-axis coordinates and H for y-axis coordinates.

This feature is an indicative of the movement of the blobs, describing how they change their position along the video. Retrieving a low value in standard calculations means that the blobs are moving almost at a constant speed. This could imply a comfortable perception for the viewer in comparison with an hectic movement.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std deviation
f_{115}	meanBiggestSpeed_x	0	1	0	0.027	0.011	0.006
f_{116}	meanBiggestSpeed_y	0	1	0.001	0.022	0.008	0.004
f_{117}	stdBiggestSpeed_x	0	1	0.017	0.116	0.07	0.02
f_{118}	stdBiggestSpeed_y	0	1	0.016	0.09	0.048	0.015

Table 3.14: **Speed** features results

As it can be seen in most of the results, mean speed values are not extremely high; supporting the standard deviation results retrieved in the *centroids* and *extrema* features.

Acceleration

Acceleration is calculated through the comparison that can be stablished between the speeds of the same blob (only the biggest for this case) in two consecutive frames

Biggest blob speeds of both frames are subtracted in pairs (x-axis and y-axis). Deceleration should be taken into account, as well as explained for the collection of features related with speed.

This feature is normalized to unity too, being the unity *W* for x-axis coordinates and *H* for y-axis coordinates.

Feature ID	Short name	Theo. min	Theo. max	Min	Max	Mean	Std
f_{119}	meanBiggestAcceleration_x	0	1	0.001	0.04	0.015	0.007
f_{120}	meanBiggestAcceleration_y	0	1	0.001	0.02	0.011	0.005
f_{121}	stdBiggestAcceleration_x	0	1	0.023	0.17	0.097	0.03
f_{122}	stdBiggestAcceleration_y	0	1	0.021	0.12	0.065	0.021

Table 15: **Acceleration** features results

Visual examples of these motion features can be visualized in chapter 5.

Chapter 4

Classification Process

After acquiring the saliency features of the video collection, a classification process can take place.

This process is the last step of the experimentation and the main goal, studying if truly the objective extracted features have an important impact in the viewer perception. In order to do so, two main steps will be followed. First of all, we need to stablish a *features selection,* reducing the number of features to work with, since the feature collection extracted from the videos is really big and maybe some of them are not enough significant in the video classification process. Different amounts and combinations of features should be taken into account, stablishing the best solution and trying to avoid overfitting or information loss.

However, it is really difficult to avoid overfitting for our research case. The database we are using; obtained from [2], was originally formed by 2732 videos. Although, only 138 of them were finally selected due implementation decisions, providing a smaller database. Despite this matter, the proposed model can perfectly accomplish its duty, but overfitting might exist, failing drastically when making predictions of new characteristics which have been unseen in the training collection.

Once the feature selection is defined, we need to be focused on the different *classification* algorithms that can be applied. We will study how different algorithms work with the different features selection we stablished at the previous step. This way, the objective is to make different combinations between the features collections and the available classification algorithms. This process is made in order to achieve the proper combination, which is able to predict successfully, as many video labels as possible, classifying them into *bad* or *good videos.*

These two steps are explained in more detail below.

4.1 Feature selection

This process is extremely important since the result can heavily differ depending on the features selected. This process allows to avoid the generalization and overfitting problems, as well as to reduce the amount of data to work with, reducing the complexity and providing an easier interpretation of the results.

In order to stablish the best logical selection of the features, WEKA software has been used. This tool allows us to stablish different feature selections, relying on a selection method that can be defined as a combination of a **search technique** with an **attribute evaluation.** The search technique looks for and proposes new feature subsets, whereas the attribute evaluation process scores every selected feature subset, allowing us to choose the proper subsets for our study. This combination is accomplished in the following steps, which are the main steps followed in order to deal with a proper feature selection.

1. **Starting point**: It is referred to the preconditions which are used to start with the selection process, having several alternatives available. On the one hand, we can start with no features, adding new ones progressively which is known as *forward selection*. On the other hand, starting with all the features, we will remove them progressively, attending to their importance in the subset which is known as *backward selection.* Finally, we have *outward selection,* referring to start the process with some of the features already selected.
2. **Search organization**: Since we have *N* features to work with, we could manage until 2^N combinations, being this assumption really difficult to deal with. In order to choose the best combination with a less tricky process, heuristic search strategies can be used providing good results. These results might not be the best ones though.
3. **Evaluation of subsets**: This step implies to find a method to evaluate the different subsets and compare them. The strategy used to evaluate the subsets is important, affecting heavily the results retrieved.
4. **Stopping point:** It is referred to the point where the process ends following certain criteria. For instance, the process might end when all the space has been covered, or until no subset improves the results adding or removing new features.

WEKA software has wrappers and filters as attribute evaluators and they can be combined with different search techniques.

WEKA not only provides a tool to compare combinations of different classification techniques with different datasets, but also allows the user to perform the K-fold cross validation algorithm on the classification experiments. Moreover it includes the option to perform the experiment a number of random iterations, testing the experiment under different circumstances and providing as consequence a more reliable result.

Before explaining K-fold cross validation algorithm, there is an important aspect to point named *validation*. Given a dataset *S* of *K* instances, we can split this collection into a training group, S_{train} with *K-N* instances and a validation group, S_{val} with *N* instances. The value of *N* should be discussed, since choosing a high value gives us a

more reliable validation, but a worse prediction due to the amount of training instances would be smaller.

In order to solve this problem, *K-fold cross-validation* is the best solution. This process is executed in rounds and involves the partitioning of the dataset in two subsets, training subset and validation subset. For each round, the analysis is performed on the training subset, while the analysis on the validation subset is validated.

Different partitions are made for each round and validation results are averaged over all the sessions, defining the error of the estimate as follows:

$$E_{CV} = \frac{1}{N}\sum_{n=1}^{N} e_n \tag{4.1}$$

This algorithm will be used for the classification process, however, for the features selection, we will work with the full training set of features and with an specific attribute evaluator, the *SVM (SVMAttributeEval),* which must be used with a *Ranker* as a search technique. This algorithm evaluates each feature independently by using a SVM (Support Vector Machine) classifier and then, ranks the *N* best ones basing on the square of the weight assigned by the *SVM*.

5,10, 15 and 20 have been chosen for *N*, stablishing different tops of features in order to study different combinations. These 20 top features are listed from top to bottom in the table described below.

N	Feature
1	stdBiggestExtremaTopRight_y
2	stdBiggestExtremaLeftTop_x
3	stdBiggestExtremaTopLeft_y
4	stdNumOfBlobs
5	stdBiggestExtremaLeftBottom_x
6	meanOverallEccentricity
7	stdBiggestOrientation
8	meanBiggestAcceleration_y
9	stdBiggestAcceleration_y
10	stdOverallExtremaLeftBottom_y
11	meanBiggestSpeed_y
12	stdBiggestSpeed_y
13	meanOverallExtremaBottomLeft_y
14	meanBiggestExtremaTopRight_y
15	stdOverallEccentricity
16	stdBiggestMajorAxis
17	stdBiggestExtremaBottomLeft_x
18	meanBiggestExtremaTopLeft_y
19	stdBiggestArea
20	stdOverallPerimeter

Table 4.1: Top 20 saliency features list

It is important to point that most of the 20 top features are related to movement and position of the blobs, i.e. *Extrema* features that specify coordinates of the blobs edges, and the features related to the speed (changes in the coordinates of the blobs due to the movement). Moreover, and concerning to this kind of features, most of them are referred to y-axis, as well as to the biggest blob. Also, 14 of these top 20 features (70%) are associated with *standard deviation* calculations against *mean* calculations.

This collection of conjectures might give us the idea that changes in the movement of the main/biggest saliency area along the y-axis, as well as in its speed are important factors when classifying correctly the videos, and as consequence, for the subjective perception of the viewer.

4.2 Classification

This step is referred to the machine learning process. Machine learning is a field of artificial intelligence, which studies algorithms able to learn from input data, extracting certain characteristics from them in order to make predictions of new incoming data. According to Arthur Samuel, “machine learning gives to the computers the capability to learn by their own without being programmed”.

It is made by two different stages: First of all, we need to train our classifier, providing an input dataset. After this, the classifier is going to be tested using the previous trained model providing a new dataset. The process is explained in more detail below.

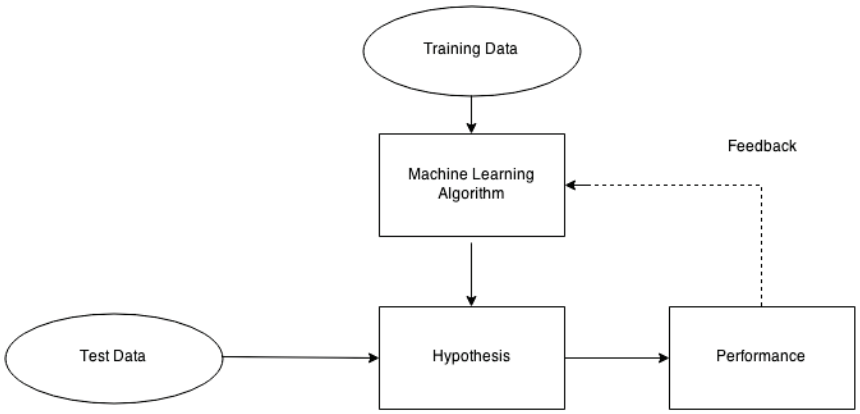

Figure 4.1: Machine learning process diagram

The machine learning process, illustrated in Figure 4.1, stablishes a hypothesis as a function between the input data set, *x* and the output of the system, *y*, i.e. *h: X→Y.* This function or hypothesis is built from an already known data set (Training Data) in which the *machine learning algorithm* has been applied. Once the hypothesis/function is built, the next step consists in assigning to each new input *x* (Test Data) a *y* output, in function of the hypothesis stablished, being the main goal the satisfactory classification of each *x* new input to the correspond *y* output.

Depending on the consistence of the hypothesis, different results will be retrieved, varying the performance of our system. This way, it is important to choose a proper training data set as well as the learning algorithm that is being applied to it.

Depending on the case, two kinds of machine learning can be performed, **classification** or **regression**

Classification deals with the problem of predicting which class of data (among an already known collection) a new instance or observation belongs to, taking as a reference, a training dataset in which every observation is already labeled. The element in charge of this task is known as *classifier.*

Regarding to classification process, there are several types of classifiers and algorithms of classification, such as linear classifiers, neural networks or decision trees among others. Next, the classifiers that are used in this research are explained:

Rule-based classifiers

- **ZeroR:** The Zero Rule algorithm is the simplest classifier we are going to use. The method is based on predicting the majority category (class), ignoring the predictors at all, i.e. the features. This way, ZeroR classifier does not predict anything; it is simply used as a baseline for other classification methods.
- **OneR:** The One Rule algorithm generates one rule for each feature, choosing the rule with the smallest total error. In order to stablish a rule for each feature, a *frequency table* must be built against the class, compiling all the combinations between the feature value and the available classes for the values (i.e., all the instances of the feature are used in the table). Thus, the total error of each frequency table is the measure of each feature contribution. The lower the error, the higher the contribution to our classification model.

Function-based classifiers

- **SimpleLogistic:** This classifier is a logistic regression model. It is directly based on a *linear classifier,* which makes the classification decision basing on the value of a linear combination of the available features. Thus, having the vector *x* as the input feature vector, the output will have the form:

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right) \tag{4.2}$$

  where $\vec{w}$ is a vector of weights obtained from the training set, assigning for each feature a specific weight and making a linear combination. This way, *f* maps each value above a certain threshold (line) to one class, whereas the values below the threshold are mapped to the other class.
  But while a linear classifier uses a line for determining classes, logistic models use a logistic curve with the form of the following function:

$$f(t) = \frac{1}{1 + e^{-t}} \tag{4.3}$$

  Moreover, the logistic regression models stablish the weight through the *gradient descent* algorithm in order to find the local minimum of the function, which minimizes the error. The implementation of this classifier in WEKA has been based on LogitBoost algorithm, which is described in the research [31].

- **Logistic:** This classifier has been implemented by WEKA generalizing the logistic regression concept. It intends to improve *SimpleLogistic* performance, adding a ridge estimator which is based on the algorithm presented by le Cessie and Houwelingen [32].

- **SMO:** Referred to *Sequential Minimal Optimization* algorithm, introduced by John Platt[33] in order to solve some problems during the training with support vector machines (SVM). A SVM consists in a linear classifier which optimizes the classification. In order to do so, if the training data set is linearly separable, two hyperplanes instead of one can be described, maximizing their distance (margin) on condition that no instances should exist between them. These hyperplanes are described by the line segment formulas:

  $$w \cdot x_i - b = 1.$$
  $$w \cdot x_i - b = -1 \quad (4.4)$$

  referring w to the weight, b to the slope and y_1=1, y_2=-1 for the classes. Thus, the distance between these two hyperplanes will be $\frac{2}{\|w\|}$, being the main goal of the algorithm to minimize the norm of the weight subject to the condition $y_i(w \cdot x_i - b) \geq 1$ for any $i = \{1, \dots, n\}$

Our experiment has been configured in such a way that 10 random iterations of 10-fold cross validation has been performed on each feature set for each of the 5 different classification algorithms we have selected.

The four feature subsets (5, 10, 15 and 20 top saliency features), formed by the top 20 saliency features described in Table 4.1, have been used with this configuration by the *Experiment* WEKA functionality, testing the performance of all the classifiers previously explained. The results are presented below

Dataset	ZeroR	OneR	Simple Logistic	Logistic	SMO
5 top	55.77	44.41	58.56	60.52	57.19
10 top	55.77	52.77	61.19	63.09	61.90*
15 top	55.77	49.46	61.93	63.35	61.75*
20 top	55.77	49.46	64.06	66.09*	63.05*

Table 4.2: Different data sets and algorithms classification results

4.3 Results discussion

Now that all the results have been extracted from all the possible combinations; combining the features subsets with the proposed classifiers, a conclusion aimed at the main goal of the research can be discussed.

Firstly, it is clear that *OneR* algorithm does not provide good results. This is an indicative that choosing the feature with the smallest total error is not enough to classify properly our videos, i.e. we need to take into account more than one feature in order to

overpass the basis stablished by the *ZeroR* algorithm. In addition, all the other algorithms; which use several features for the classification; improve the results obtained by *ZeroR* algorithm.

On the other hand, as explained in the previous section, *Logistic* algorithm was performed in order to improve *Simple Logistic* performance, being this fact perfectly observable in the results retrieved. Finally, working with *Logistic* algorithm and using the top 20 saliency features as the dataset, we achieve the best result in comparison with other algorithms and datasets combinations.

4.3.1 Statistical significance

Despite the result is better than the basis, we cannot guarantee that this improvement is due our classifier is actually learning, because it might be the result of a random chance. In order to measure this issue somehow, we need to prove the *statistical significance* of the result. A result is statistically significant if the probability to be achieved by chance is low, existing statistic evidences of the improvement.

The significance of a test is concerned with the verification of a hypothesis denoted as *null-hypothesis,* H_o. More simply, it is defined as the probability p to reject this hypothesis, implying that some errors might happen as explained in [34]. These errors are classified in two types, Type I error and Type II error. Type I error (expressed by α) happens when H_o is wrongly rejected when it is in fact true, whereas Type II error occurs when H_o is not rejected when it is in fact false.

This way, if a statistical hypothesis testing provides a p value lower than α (which it most often set at 0.05 (5%)), the *null-hypothesis* is rejected, being this result statistically significant, which is formally written as $p< 0.05$. The lower the α value, the stronger the evidence that a result is statistically significant.

However, there is a dilemma concerning α, since low values provide more statistical significance but also takes the risk that Type II error happens, losing *power* of the test. For this case, it could be appropriate to increase the sample size of the study, increasing the power too and reaching the significance level.

An important statistical test based on hypothesis is the *t-test* in which a Student's t distribution is followed, stablishing the null-hypothesis, H_o as not rejected. This concept was introduced by William Sealy Gosset in 1908, and can determine if two sets of data are statistically different from each other, relying on sample means comparison, that is, the application that has been used for this research.

Depending on the conditions, the nature of the samples and the application, several types of t-test can be implemented. Next, three different types of them are discussed [34].

- One-sample t-test: This t-test is used for doing a comparison of a single sample mean with a fixed value of interest, named "gold standard", which is totally independent of the sample. The main goal of this test is to determine if there is enough statistical evidence in order to conclude that the mean of the population related to the sample, is different from this fixed value. Depending on the assumption taken for the null-hypothesis, one-sample t-test can be performed as one- or two-tailed test.
- Two-sample t-test: This time, the comparison is stablished between two population means based on independent samples (i.e., unrelated to each other). Hence, if the independent sample means are significantly different, it can be concluded that the mean of both populations the samples were extracted from, are statistically different from each other. As well as the one-sample t-test, one- or two-tailed test can be performed depending on the assumption regarding with the null-hypothesis.
- Paired t-test: This test is similar to two-sample t-test but the samples to compare are related or paired some way. Thus, the data to be studied is focused on the difference within the pair, calculating difference scores which will be analyzed as a one-sample t-test.

As previously commented, the conditions and characteristics of the experiment, determine which t-test should be performed. For this research, we will be focused on the average accuracy of the proposed classifiers in comparison with a baseline classifier. Thus, the samples to compare are referred to all the accuracy results that the proposed classifiers provide in the experiment, taking into account each iteration and each fold of the cross validation over the different feature subsets. Since the experiment should be performed in pairs and these samples are related to each other; because they share the same feature subsets, a paired t-test should be executed for different pairs of classifiers in which one of them should be treated as a baseline classifier. Among the classifiers previously described, the ZeroR classifier will be the baseline for all the other ones.

The paired t-test is computed, obtaining the difference between the scores of both samples (classifiers for our case), which forms a single distribution X_D. Through this single distribution, some statistical measures can be obtained such as the mean $\overline{X_D}$ and the standard deviation, s_D. Moreover, the mean of differences, μ_0, is usually treated as the null-hypothesis for this test (H_o). With all this information and defining n as the sample size, the t-statistic value of the test can be calculated using the Equation 4.5

$$t = \frac{\overline{X_D} - \mu_0}{s_D / \sqrt{n}} \tag{4.5}$$

From the point of view of this research, the greater the value of t, the greater the rejection against the null-hypothesis, confirming this way the existence of significant difference between the proposed classification experiment and the one that uses the baseline classifier, ZeroR. Using this value and taking into account the degrees of freedom (*n-1* for this case) the p-value can be derived.

4.3.2. Results conclusions

Taking into account the concept explained in the previous section, statistically significant results are marked by "*" in Table 4.2. As it can be seen, despite some algorithms overpass the performance of the basis, they should not be considered as statically significant, such as Simple Logistic algorithm.

Despite most of the statistically significant results rely on the SMO algorithm, Logistic algorithm provides the best result, but only combining it with the top 20 saliency features, discarding the rest of the combinations due to statistical significance absence.

With this combination of algorithm-data set, it can be affirmed statistically that the implemented classifier is able to classify properly the 66.09% of new unlabeled videos, basing its classification on the extracted saliency features from the 138 videos which form the training set.

In conclusion, it has been demonstrated that saliency can be considered an important field of study concerned with video assessment, having an influential impact in the subjective perception of videos.

Chapter 5

Analysis of experiments

After the results discussion, an experiment based on them and their respective videos is proposed.

In order to start with the experiment, the top twenty saliency features have been analyzed, studying for each feature one video per class. These videos will have the most significant value of the feature in question among all the videos of their class, using these videos as comparison examples for the feature study. So, we can analyze how characteristic each feature is for its class, as well as measure somehow their contribution in their classification process.

For instance, the video number with id 73 extracted from the collection can be used as a clear example of a video belonging to bad videos. This affirmation relies on some of its features values, since they are the most significant ones in comparison to the other videos' of the same class. This way, video number 73 has the lowest value for the features *stdNumOfBlobs*, *meanBiggestAcceleration_y*, *meanOverallExtremaBottomLeft_y,* *meanBiggestExtremaTopRight_y* and *meanBiggestExtremaTopLeft_y* in comparison with the other videos of the same class. Furthermore, these features are ranked among the top fifteen features, adding certain relevance to them.

On the other hand, video with id 122 can be used as an example of a video belonging to good videos. This video has the lowest value for the features *stdBiggestOrientation*, *stdBiggestMajorAxis*, *stdBiggestArea* and *stdOverallPerimeter* and also the highest for the features *meanOverallExtremaBottomLeft_y*, *meanBiggestExtremaTopRight_y* and *meanBiggestExtremaTopLeft_y*. In addition, these features are ranked among the top twenty features, adding certain importance as well.

In order to verify in practice the impact of these top twenty features when classifying the videos, a comparison study is detailed further. To do so, some frames and graphs extracted from the videos with significant features values, are going to be compared attending to their class. This study is focused on the blobs of these frames verifying their correct behavior and aspect in comparison with their respective retrieved values. This way, we can verify visually the fundamental impact of these features in the classification process.

Therefore, this experiment is going to be stablished using a pair of videos for each feature, i.e. the videos with the most significant value of the studied feature, in comparison with all the videos of their respective class. In other words, we collect extreme values belonging to videos of different classes for the same feature. Once this step has been accomplished, these extreme values are compared along the frames of both videos, by plotting a graph where the x-axis measures the frames of the video and the y-axis the value of the feature. Below, some of the clearest examples are explained. Note that no video has been repeated as example when comparing the feature values, giving more validity to the study case of how the feature values can describe the class of a video. The features that are about to be compared have been split in two groups, **shape** and **movement.**

5.1 Shape analysis

Regarding the shape, the first feature to be compared is the major axis of the ellipse that can be described in the biggest blob of the frame in question. In order to study this feature, the videos with id 135 and 122 belonging to class bad videos and to class good videos respectively are compared. This comparison can be seen in Figure 5.1, in which this major axis length seems to be more changeable in video with id 135 than the one related to video 122.

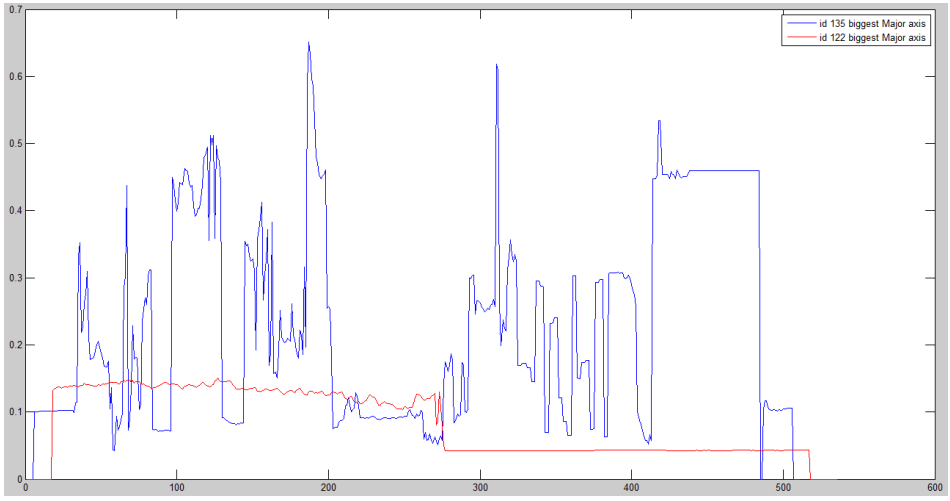

Figure 5.1: **Biggest Major Axis** feature comparison for videos 122 and 135

Furthermore, in video 122 this feature is constant along almost 200 frames whereas in video 135 the feature does not stop changing. This is because video 122 is totally static on the camera movement, displaying only one object (the car, which is moving rotationally) ,whereas video 135, in spite of the camera does not move a lot, many scenes with different points of view are displayed for few seconds, changing the attention spot constantly. Some frames extracted from these videos are shown below, highlighting the saliency area. Whereas in video122, the salient region is almost

constantly centered in an advertisement related with the car and displayed in the corner, in video 135 salient region is continuously changeable, highlighting different regions.

Figure 5.2: Frame 285 and 517 belonging to video 122

Figure 5.3: Frame 83 and 103 belonging to video 135

The same happens with the pair of videos 4 and 134 belonging to class bad videos and good videos, respectively. The camera of video 134 is totally static as the car and the saliency areas are focused on it, despite some objects are in movement (people entering and exiting from the car).

On the other hand, video 4 despite is not as hectic as video, is constituted by several scenes, displaying some famous people at the beginning of the video and the car at the end. As consequence, saliency areas are more changeable in video 4 as it can be seen in Figure 5.4 in which the feature *overallEccentricity* is compared for video 4 and 134.

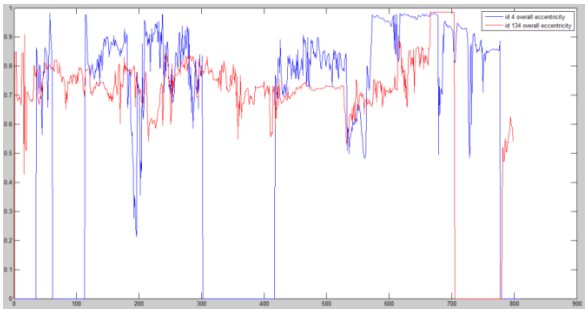

Figure 5.4: **Overall Eccentricity** feature comparison for videos 4 and 134

5.2 Movement analysis

Regarding the movement, the first feature to be compared is the speed of the biggest blob. It is important to point that this biggest blob does not have to be referred to the same object in the video, since the scene might change. For this case, videos with id 52 and 13 belonging to class bad videos and to class good videos respectively, are compared, providing the results that can be checked in Figure 5.5 shown below.

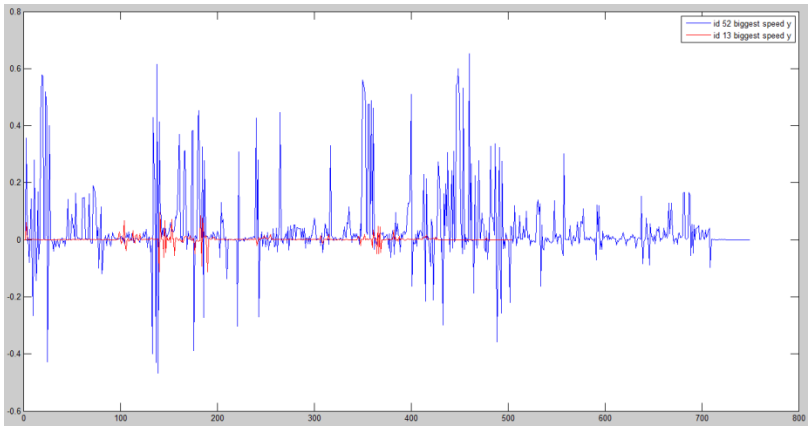

Figure 5.5: **Bigges Speed y** feature comparison for videos 52 and 13

As previous good videos examples, in video 13, the camera barely moves, locating the saliency area at the center of the screen. As consequence, biggest blob speed is insignificant. On the other hand, video 52 is even more hectic than video 135, displaying a spark constantly moving through a dark background (which makes easier to identify the spark as the biggest blob). As consequence, biggest blob speed for this "bad" video will have high values, as it can be seen in the frames shown below. It is important to point that these frames are consecutive in the video stream, adding importance to the high speed value of the blob.

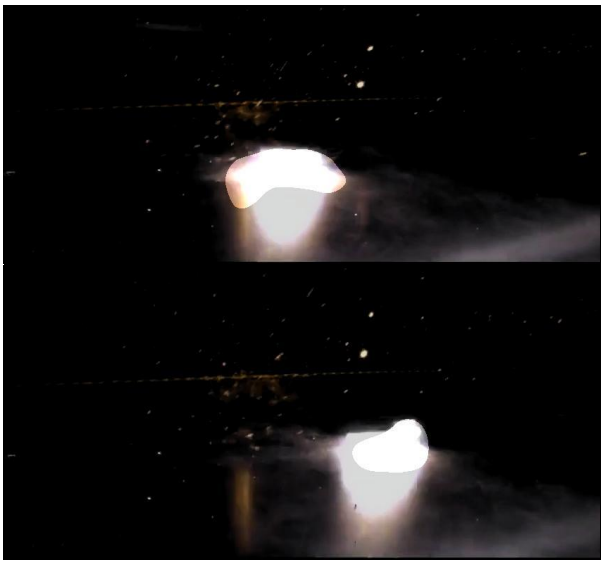

Figure 17: Frames 442 and 443 belonging to video 52

As well as the speed, video 52 has the highest value for the feature *stdbiggestAcceleration* among all the videos of its class, but there are more features to compare regarding the movement of the blobs, for instance, changes in the coordinates related to the blobs.

For this case, the value of the feature *stdBiggestExtremaBottomLeft_x* is extremely different for videos 91 and 18, belonging to class bad videos and to class good videos respectively.

Again the video belonging to the bad class is more dynamic, representing several fast scenes with camera movements, changing the position of blobs for each scene, i.e. constantly. On the other hand, despite it has several scenes as well, video 18 is more static, since its scenes are displayed for a longer time with no camera movements. As consequence, position of the blobs are the same for long periods, and only changes when new scenes are displayed (It is important to point that some of these scenes are displayed more than once). The evolution of the feature value for both videos can be observed in the figure described below.

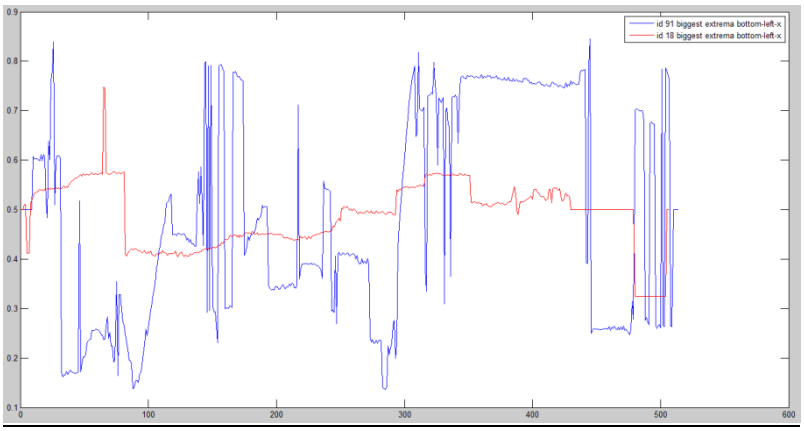

Figure 18: **Biggest Extrema Bottom-Left-x** feature comparison for videos 91 and 18

In conclusion, videos belonging to good class seem to be more static and quite whereas bad class videos seem to be more dynamic. This affirmation is taken as theoretical though, because only some videos have been compared for this study, being the comparison video by video out of the scope for this research.

This experiment supports the assumption that static and quite videos where the attention spots (or salient regions) are static, seem to be more comfortable for the viewer, causing a good impression and enhancing its perception. On the other hand; hectic videos where the salient regions are constantly in motion, seem to be less comfortable for the viewer, causing a negative perception.

Chapter 6

Conclusions

The main objective of this chapter is to make a short brief of the research and to discuss the most important conclusions obtained after its implementation. In addition, the contributions to the state of the art concerning with saliency will be remembered as well as the new saliency features proposed in order to achieve the objects of this research, developing a model for the automatic assessment of viewer subjective perception of Youtube videos.

Finally, some feature work will be mentioned, incorporating new fields and studies for the research.

6.1 Brief and main conclusions

In this research, we have presented a new unsupervised learning method based on saliency for car commercial videos extracted from the video platform YouTube. For this purpose, we have used the same video dataset as the used in [2], formed by 138 car commercial videos which were divided into two classes, *bad* and *good* videos, basing this division on a cluster analysis performed in this research.

Starting from this point, a saliency map was extracted from each frame of the videos using the GBVS algorithm [13] and simplifying the map through the application of a threshold. Then, a collection of 15 types of saliency features has been extracted from all the simplified saliency maps related to the frames, computing in total 122 saliency features for each video when dealing with statistical measures.

Then, several classification experiments have been executed using different proposed classifiers and different datasets based on a feature selection algorithm. Finally, these models have been compared achieving a classification accuracy of 66.09% as the best result. Furthermore, it has been demonstrated that the result is statically significant, concluding that saliency does affect the subject perception of videos.

This way, some important contributions have been made with this research:

- Regarding with the main goal of the research, it has been concluded that saliency does modify the subject perception of videos. In addition, contributes to the state of the art of this field, incorporating a new domain, car commercial videos.
- The database used for this research has been obtained from [2], extending this research by incorporating the concept of saliency and reinforcing the idea of the existence of objective data implication in video quality assessment.
- A new collection of saliency features has been stablished, demonstrating their usefulness using the proposed model when dealing with saliency, a concept that

is still in exploration. Furthermore, this collection models dynamic saliency aspects and takes into account the saliency evolution along the frames with features concerning with motion.

6.2 Future work

Several extensions can be applied for this work. We have been focused on the saliency concept but others features related with other fields can be added to the collection.

- **Study of the correlation of saliency and visual features with the impression perceived by the viewer:**
  Keeping the interest of studying the saliency concept, visual features can be added, extending this way the collection for our model. These features were extracted in [2] and in addition, their usefulness was perfectly demonstrated.
- **Study of the correlation of saliency and audio features with the impression perceived by the viewer:**
  Concerning with auditory perception, audio features can be added to the collection. In this way, it would attempt to compile both saliency and audio features trying to correlate them with the impression perceived by the viewer.
- **Application of this model to other video domains**

  We have proved the usefulness of this model when working with car commercial videos as the main domain. Although, applying this model to other video domains could be an interesting idea, seeing how the results may change depending on the content that is being analyzed.

- **Video compression improvement**
  As it has been discussed along this research, the techniques and the proposed solutions retrieve good results but imply a high computational cost. Despite this is not a primary goal for an experimentation research, computational cost could be reduced using video compression, lessening the information to work with.

Appendices

WEKA (*Waikato Environment for Knowledge Analysis*) is a machine learning free software developed in Java language at the University of Waikato, New Zealand. It provides a collection of tools concerning with data analysis and predictive modelling supporting preprocessing, classification, clustering, regression and feature selection algorithms among other tasks.

Explorer and *Experimenter* are the WEKA tools that have been used for this research. On the one hand, *Explorer* allowed us to select the different data sets we worked with through a search technique and an attribute selector as exposed in chapter 5. On the other hand, *Experimenter* provides the necessary tools to test different classification algorithms on several data sets, obtaining the results of the process.

In order to work with this software, .arff files have been created with the information collected from the MATLAB code. Next, .arff files structure is explained in more detail.

ARFF [35] is the file format used for WEKA software to interpret input and output data. It is formed by a list of instances sharing a same collection of attributes in such a way that the structure of the files is as follows:

Header: The header is formed by a *file identifier* and a list of the *attributes* shared by the instances detailed in the body.

- **File identifier**: A string that specifies the file name according to the following format:

  @relation <file_name>
- **Attributes:** List of the attributes shared by the instances, preceding each attribute with the keyword @attribute. In addition, each attribute must have defined its datatype having the following format:

  @attribute <attribute_name> <datatype>

In our case, all the attributes have *numeric* as datatype in exception of the attribute *video_class* having {cluster0,cluster1} as datatype in order to differentiate the classes the instances belong to.

Body: The body is formed by the list of instances separating their feature values with commas. The instances are listed by rows defining the feature values by columns. At the beginning of the body, the keyword @data should be written.

The structure commented above can be checked in Figure A.1 illustrating the arff that contains the top 20 saliency features.

```
@relation features

@attribute stdBiggestExtremaTopRight_y numeric
@attribute stdBiggestExtremaLeftTop_x numeric
@attribute stdBiggestExtremaTopLeft_y numeric
@attribute stdNumOfBlobs numeric
@attribute stdBiggestExtremaLeftBottom_x numeric
@attribute meanOverallEccentricity numeric
@attribute stdBiggestOrientation numeric
@attribute meanBiggestAcceleration_y numeric
@attribute stdBiggestAcceleration_y numeric
@attribute stdOverallExtremaLeftBottom_y numeric
@attribute meanBiggestSpeed_y numeric
@attribute stdBiggestSpeed_y numeric
@attribute meanOverallExtremaBottomLeft_y numeric
@attribute meanBiggestExtremaTopRight_y numeric
@attribute stdOverallEccentricity numeric
@attribute stdBiggestMajorAxis numeric
@attribute stdBiggestExtremaBottomLeft_x numeric
@attribute meanBiggestExtremaTopLeft_y numeric
@attribute stdBiggestArea numeric
@attribute stdOverallPerimeter numeric
@attribute video_class {cluster0,cluster1}

@data
0.1117,0.1821,0.1117,1.7977,0.1821,0.5866,46.7571,0.0117,0.0657,0.0875,0.0086,0.051,0.5214,0.4058,0.3032,0.0633,0.1861,0.4058,0.008,0.1462,cluster0
0.1202,0.1733,0.1202,0.8663,0.1733,0.7641,49.3172,0.0107,0.0746,0.1112,0.0078,0.0497,0.6034,0.3999,0.1573,0.0788,0.153,0.3999,0.0151,0.1106,cluster0
0.1281,0.1668,0.1281,0.8315,0.1668,0.7192,48.1502,0.0109,0.0696,0.1023,0.0075,0.0487,0.5995,0.4122,0.1886,0.0802,0.1679,0.4122,0.0112,0.0988,cluster1
0.1105,0.1616,0.1105,1.6473,0.1616,0.5282,32.3474,0.0197,0.0671,0.0871,0.0172,0.06,0.5111,0.4353,0.41,0.0897,0.1474,0.4353,0.0123,0.1538,cluster0
0.1184,0.1622,0.1184,0.8704,0.1622,0.7413,43.0961,0.0161,0.1056,0.1074,0.0131,0.0734,0.5502,0.4538,0.1465,0.0593,0.1576,0.4538,0.0064,0.0728,cluster0
0.1106,0.1577,0.1106,1.3426,0.1577,0.5305,38.6175,0.0125,0.0652,0.0909,0.012,0.0516,0.5434,0.4553,0.3628,0.0833,0.1644,0.4553,0.011,0.1312,cluster0
0.1264,0.1874,0.1264,1.1702,0.1874,0.7036,58.6937,0.0097,0.0827,0.1024,0.0074,0.0562,0.5749,0.4019,0.2133,0.0696,0.1817,0.4019,0.0153,0.1485,cluster1
0.1165,0.143,0.1165,1.5692,0.143,0.533,45.3666,0.0074,0.0636,0.0909,0.005,0.0447,0.5612,0.4413,0.3642,0.0673,0.1461,0.4413,0.0087,0.1505,cluster0
0.114,0.1534,0.114,1.1638,0.1534,0.6976,48.2671,0.0028,0.0231,0.069,0.0023,0.0202,0.5863,0.433,0.148,0.0597,0.1354,0.433,0.0144,0.1544,cluster0
0.1236,0.1549,0.1236,1.207,0.1549,0.6697,44.3341,0.0104,0.0605,0.1021,0.0091,0.0468,0.5562,0.3878,0.266,0.0956,0.1467,0.3878,0.0181,0.1545,cluster0
0.1408,0.1412,0.1408,1.2797,0.1412,0.7546,50.0298,0.0178,0.0981,0.1122,0.0136,0.0717,0.5308,0.4174,0.1508,0.0732,0.1455,0.4174,0.0096,0.1241,cluster0
0.1001,0.1236,0.1001,1.2441,0.1236,0.4688,43.0977,0.0057,0.052,0.0904,0.0037,0.0351,0.5834,0.4496,0.3855,0.0934,0.1216,0.4496,0.0153,0.1334,cluster0
0.0536,0.1511,0.0536,1.0389,0.1511,0.7024,48.7277,0.0016,0.0268,0.036,0.0012,0.0161,0.5866,0.4471,0.1798,0.0459,0.1549,0.4471,0.0079,0.0995,cluster1
0.1086,0.1657,0.1086,1.2226,0.1657,0.7664,43.0322,0.0191,0.0742,0.1085,0.0133,0.0534,0.5326,0.3577,0.1511,0.0799,0.1663,0.3577,0.0165,0.1356,cluster0
0.0934,0.1079,0.0934,1.2459,0.1079,0.4203,36.7972,0.0096,0.0602,0.0508,0.0072,0.048,0.5395,0.4554,0.3891,0.1146,0.0996,0.4554,0.0269,0.1471,cluster1
0.1313,0.1719,0.1313,0.8535,0.1719,0.6034,48.7481,0.0064,0.0641,0.1043,0.0065,0.044,0.5454,0.4036,0.3173,0.0754,0.1806,0.4036,0.0111,0.0896,cluster1
```

Figure A.1: Top 20 saliency features arff file

Next, some pseudo-code belonging to the main part of the code is detailed. It is referred to the main tasks of this research dealing with the saliency map and features extraction as well as the processes that have been carried out on them. The pseudo-code referred to the post process that has been used to perform the comparison study of the chapter 6 is detailed too.

Main process code

Input args: N: Number of frames of the video, I: Image extracted for each frame of the video

Output args: *struct*: Struct containing information regarding the video, *metrics:* Array containing the feature values

Initialisation and preprocess;

for *n=1 to N* **do:**

$DIF\_RG = \sum(\sum(R\ layer\ of\ I - G\ layer\ of\ I))$

$DIF\_GB = \sum(\sum(G\ layer\ of\ I - B\ layer\ of\ I))$

if (DIF_RG ≠ 0 AND DIF_GB ≠ 0) **then**

% I is not a black image

Gbvs saliency map extraction, *map*;

Height and width extraction from map, *H, W;*

%Apply 0.75 threshold to the map

for *w=1 to W* **do:**

for *h=1 to H* **do:**

if*(map(w,h)>threshold)* **then**

thresholdMap(w,h)= 1;

else

thresholdMap(w,h)= 0;

end

end

end

Extraction of connected components from *thresholdMap* with *bwconncomp* function;

Extraction of features for each connected component (*blob*) with *regionprops* function;

for *i=1 to number of blobs* **do:**

areas(i) = blob(i).Area

end

biggestBlob = blob with max(areas);

for each *feature* **do:**

for *i=1 to number of blobs* **do:**

if *blob(i) = biggestBlob* **then**:

biggestBlobFeature = blob(i).feature;

Normalisation of *biggestBlobFeature*;

end

allBlobsFeature(i) = blob(i).feature_name;

Normalisation of *allBlobsFeature(i);*

end

overall calculation through *allBlobsFeature(i)*

end

Saving process for frame features (*frame(n))*;

else

% I is a black image

Feature values regarding coordinates = 0.5 (Center of the image);

Rest of feature values = 0;

end

mean and *std* calculations of all the frame features of the video through *frame(n).*

Construction of *struct* struct and *metrics* array mentioned in chapter.

Post-processing code

Input args: N: Number of frames of the video, W: Width of the image, H: Height of the image.

Output args: Returns the same image, highlighting the saliency area modifying the image brightness

for *n=1 to N* **do:**

Conversion of the image into hsv format;

Splitting process of hsv image retrieving *H*,*S* and *V* layers;

for *w=1 to W* **do:**

for *h=1 to H* **do:**

if*(map(w,h)=1)* **then**

V(w,h) = V(w,h) + 0.3;

else

V(w,h) = V(w,h) - 0.15;

end

end

end

Linking process of the layers *H,S* and new *V*, forming the new hsv image;

Conversion of the hsv image into rgb format;

Saving of the new image;

Once all the features have been extracted for each frame, all the information is stored and compile with the extraction of the same features for the rest of the frames of the video. After extracting all the frames information, we will compile this information

building a struct named as *<VID_ID>_struct* for each video where <VID_ID> is referred to the ID of the video.

Struct: The struct contains the fields *name*, *frames*, *metrics* and *features*

-Name*:* Contains the ID (name) of the video

-Frames*: num_frames* X 1 structs array. The structs are referred to the frames of the video. Each one contains information about the concerning frame (Name of the frame, size of the file, creation date, among others)

-Metrics: *num_frames* X 1 structs array. The structs are referred to the frames of the video. Each one contains two structs more. The first one contains the saliency binary map of the image whereas the second one the frame feature values for the concerning frame

-Features: 1 X 122 array containing the video feature values

In addition, a *num_frames* X 61 array named as *<VID_ID>_metrics* is built for each video too, in order to compile all the frame feature values

References

[1] Bar-Haim, Y., Lamy, D., Pergamin, L., Bakermans-Kranenburg, M.J., van IJzendoorn, M.H.: Threat-related attentional bias in anxious and non-anxious individuals: A meta-analytic study. Psychological Bulletin (2007).

[2] Hernández, A.: Aesthetics Assessment of Videos through Visual Descriptors and Automatic Polarity Annotation. Universidad Carlos III de Madrid (2014).

[2] James, W.: The Principles of Psychology, Vol. 1. Dover Publications (1950).

[4]. Itti, L.: Automatic foveation for video compression using a neurobiological model of visual attention. IEEE Transactions on Image Processing 13 (2004) 1304-1318.

[5]. Bruce, N., Tsotsos, J.: Saliency based on information maximization. Advances in neural information processing systems 18 (2006) 155.

[6]. Seo, H., Milanfar, P.: Static and space-time visual saliency detection by selfresemblance. Journal of Vision 9 (2009) 1-12.

[7]. Zhang, L., Tong, M., Marks, T., Shan, H., Cottrell, G.: Sun: A bayesian framework for saliency using natural statistics. Journal of Vision 8 (2008) 1-20.

[8] Koch, C., Ullman, S.: Shifts in Selective Visual-Attention-Towards the Underlying Neural Circuitry. Human Neurobiology 4 (1985) 219-227.

[9] Niebur, E., Koch, C.: Control of Selective Visual Attention: Modelling the “Where” Pathway (1996).

[10] Cerf, M., Frady, E. P., Koch, C.: Faces and text attract gaze independent of the task: Experimental data and computer model, Journal of Vision 9 (2010) 1–15.

[11]Posner, M., M. I.: Orienting of attention. The Quarterly journal of experimental psychology 32 (1980) 3–25.

[12] Itti, L., Koch, C., Niebur, E.: A Model of Saliency-Based Visual Attention for Rapid Scene Analysis IEEE Transactions on Pattern Analysis and Machine Intelligence 20(11):1254-1259.

[13] Harel, J., Koch, C., Perona, P.: Graph-Based Visual Saliency. Proc. Advances in Neural Information Processing Systems (2007) 681-688.

[14] Seo, H., Milanfar, P.: Static and space-time visual saliency detection by selfresemblance. Journal of Vision 9 (2009) 1-12

[15] Culibrk, D., Mirkovic, M., Zlokolica, V., Pokric, M., Crnojevic, V., Kukolj, D.: Salient motion features for video quality assessment. Image Processing, IEEE Transactions on (2010) 1-1

[16] Mancas, M., Riche, N., Leroy, J., B.G.: Abnormal motion selection in crowds using bottom-up saliency. In: IEEE ICIP. (2011)

[17] Farnebäck G.: Two-Frame Motion Estimation Based on Polynomial Expansion, Lecture Notes in Computer Science (2003)

[18] VanRullen, R.: Visual saliency and spike timing in the ventral visual pathway (2003).

[19] Parkhurst, D., Law, K., Niebur, E.: Modelling the role of salience in the allocation of visual selective attention. Vision Research 42 (2002) 107-123.

[20] Parkhurst, D., Niebur, E.: Variable resolution displays: a theoretical, practical and behavioral evaluation. Human Factors 44 (2002) 611-29

[21] Courty, N., Marchand, E.: Visual perception based on salient features. Proc. of 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems. (2003)

[22] Hillaire, S., Breton, G., Ouarti, N., Cozort, R, Lécuyer, A.: Using a Visual Attention Model to Improve Gaze Tracking Systems in Interactive 3D Applications In Computer Graphics Forum (CGF) 29 1830-1841.

[23] Su, S. L., Durand, F., Agrawala, M. An Inverted Saliency Model for Display Enhancement. In Proceedings of 2004 MIT Student Oxygen Workshop, Ashland, MA (2004)

[24] Grigorios, K., Georgios, T.: Image based Monument Recognition using Graph based Visual Saliency. 12 (2013) 88-97.

[25] K, Rapantzikos, N, Tsapatsoulis, Y, Avrithis, S, Kollias, Spatiotemporal saliency for video classification. Signal Processing: Image Communication 24 (7), 557-571

[26]Lv, X.; Zou, D., Zhang, L., Jia, S.: Feature coding for image classification based on saliency detection and fuzzy reasoning and its application in elevator videos. WSEAS Transactions on Computers 13 (2014) 266

[27] Malik, J., Perona P.: Preattentive texture discrimination with early vision mechanisms. Journal of the Optical Society of America (1990)

[28] Bruce, N., Tsotsos, J.: Saliency Based on Information Maximization (2005)

[29] Itti L., Baldi P. Bayesian Surprise Attracts Human Attention (2005)

[30] Cerf, M., Harel, J., Einhauser, W., Koch C.: Predicting human gaze using low-level saliency combined with face detection, *Neural Information Processing Systems* (NIPS) 21 (2007)

[31] Sumner, M., Frank, E., Hall, M.: Speeding up logistic model tree induction. In 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, pages (2005) 675-683

[32] Cessie, S., Van Houwelingen, J.C.: Ridge estimators in logistic regression. Applied Statistics, 41 (1992) 191-201

[33]Platt, J.: Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning. MIT Press (1998).

[34] Allan, C., Elliot, Wayne, A: Woodward. Statistical Analysis: Quick Reference Guidebook. Sage Publications, Inc., 1 edition (2007).

[35] New Zealand University of Waikato. Attribute-relation file format (arff). http://www.cs.waikato.ac.nz/~ml/weka/arff.html, July 2013